Implementation of an Intelligent Tutoring System

IFT-6890

SINADINAKIS EMMANOUIL
SINE 241187 03

6/20/2011

Internship supervisors:

M. Pierre Chalfoun, PhD candidate & Professor Claude Frasson, Director of the HERON Laboratory
Table of Contents

1. Introduction ........................................................................................................................................... 4
2. Background / Problem ............................................................................................................................ 5
3. Project Objectives .................................................................................................................................. 7
   3.1 Designing the architecture of MAS ................................................................................................. 7
   3.2 Implementing MAS for an ITS system ............................................................................................ 8
4. Timeline .................................................................................................................................................. 9
5. Work environment .................................................................................................................................. 10
   5.1 - Overall ITS structure .................................................................................................................. 10
   5.2 - TROPOS methodology ............................................................................................................. 11
   5.3 - GUI platform ............................................................................................................................. 13
   5.4 - JADE language .......................................................................................................................... 14
   5.5 – Agent programming ................................................................................................................... 15
   5.6 - XML file handling ....................................................................................................................... 17
6. Technical Problems and their Resolutions ............................................................................................. 19
7. Critical Evaluation .................................................................................................................................. 21
8. References .............................................................................................................................................. 22
List of figures and tables

Figure 1: HERON Lab chart ...................................................................................................................... 5
Figure 2: Example of TROPOS methodology between Buyer-Seller agents, TAOM4E environment ..... 7
Figure 3: Sequenced diagram of one scenario in the ITS ........................................................................... 11
Figure 4: Goals of ITS system, in TROPOS methodology ......................................................................... 12
Figure 5: Early requirement of goal 1, in TROPOS methodology ................................................................... 13
Figure 6: GUI platform ............................................................................................................................ 13
Figure 7: Register in YP ............................................................................................................................ 14
Figure 8: Deregister from YP ..................................................................................................................... 15
Figure 9: Searching the YP for the number of existing agents .................................................................... 15
Figure 10: Cyclic behaviour of the Interface agent ...................................................................................... 16
Figure 11: Example of Instruction.xml, an XML file .................................................................................. 18
Figure 12: Filter an XML file ...................................................................................................................... 20
Figure 13: Build an XML File ..................................................................................................................... 20

Table 1: Timetable of my Internship ........................................................................................................... 9
Table 2: Sequence of exchanged messages between agents ......................................................................... 17
1. Introduction

Learning is a very complicated procedure and the precise way in which people gather and store knowledge is not yet fully understood. Every learner is almost considered as a unique “case”, or individual, because of the many differences that may exist between his/her knowledge, skills and expertise when compared regards to other learners.

In fact, traditional education systems for more than 6000 years have been focused around learning in a group setting. From primary to high school, all the way to university, the lessons take place in groups (classes) without much emphasis on the different kinds of difficulties, skills or abilities that every individual may have and must face alone. Intelligent Tutoring Systems hope to change this idea by enabling computers to properly teach “as a human” in a one-to-one setting with the student, thus building an interactive environment tailored for every individual. These interactive environments are capable of providing intelligent and individualized coaching solution and they can evaluate each learner’s actions by giving a feedback for obtaining deep and long-lasting knowledge.

Intelligent Tutoring System (ITS) is a computer system that is capable of taking decisions independently on behalf of its user or owner by following pedagogical teaching strategies. Figuring out what needs to be done to satisfy design objectives, giving back direct customized instruction or feedback to student for accomplishing a specific task. “Intelligent Tutor Systems have been around since the late 1970s, but increased in popularity in the 1990s”. [1]

There are many ways to implement an ITS. We decided in this internship to implement our ITSs by using the intelligent agents built within a multi-agent architecture providing an important advantage over existing object-oriented technologies. The main advantage of intelligent agent architecture is that agents follow standardized communication protocols and languages (FIPA, ACL Messages, yellow pages, etc) and are autonomous. Agents may communicate easily with other agents by sending messages without requiring the address or the specific methods available to them as is requiring in the object-oriented technologies. Another important advantage of agents over software programs is the ability of an agent to introspect on its beliefs and desires and act as a human would by using AI approaches to learn and adapt to each situation accordingly.

Our ITSs will consist of five different modules: the Interface, the Tutor, the Expert, the Learner and the Database. Every module has different goals to achieve. Each module will be implemented as an agent that we will implement in our system. Every agent does specific actions depending on other agents, figuring out what needs to be done to satisfy design objectives, rather than constantly being told what to do in advance. Indeed, an agent is in a close coupled continual interaction with its environment and acts according to specific perceived precepts. Interactions are dynamic and change over time. To successfully interact, agents will require the ability to cooperate, coordinate, and negotiate with each other. Multi-agent systems fulfil these aspects. In next section we will give more details about these modules and their interactions.

In this internship, I will also assist in implementing agents in a multi-agent framework (JADE) by following an agent methodology (TROPOS), in JAVA language for the construction of an Intelligent Tutoring System.
2. Background / Problem

I am member of SEPICS at the University of Montreal. The time schedule of my internship is for 4 months, starting from the 1st of March and ending at the 30th of June 2011.

Student Exchange Program In Computer Science (SEPICS) aims in initiating collaboration amongst European and Canadian universities in the field of Intelligent Systems in Computer Science, strengthening already existing separate co-operations of the European and Canadian consortiums.

The internship is performing at the HERON laboratory. HERON (Higher Educational Research On tutoring systems) is a multi-disciplinary group that includes researchers from several universities: the University of Montreal, UQAM, McGill University and UQAC. It incorporate about 41 people (4 professors and 30 students, 4 support staff). The figure below is showing the chart of HERON laboratory in Montreal University where my internship takes place. In this chart, we can see that the lab has a director Professor Claude FRASSON and a codirector Professor Esma AIMEUR. My immediate supervisor is M. Pierre CHALFOUN (PhD candidate and research assistant).

![HERON Lab chart](image)

We have mainly focused in this internship in Intelligent Tutoring System (ITS). Intelligent Tutoring System (ITS) is the result of a conjunction of work and progress in artificial intelligence and education. Scientific developments and the advancement of knowledge is part of a long tradition of research at Montreal University.

I am currently taking part in a two-phase project implicating two internship students, each with a specific task. The first phase of the project is to design the multi-agent architecture as well as the agent platform to do the intelligent/adaptive teaching. The second phase of the project is to create a
graphical user interface to communicate directly with the learner. My task focuses only on the first phase: help design the architecture (theoretical phase) and program the agents in the JADE platform (practical part). The two phases of the project come together when the learner must interact with the ITS. The system should assess, in real-time, the current progress of learner by tracking student’s performance, sending appropriate feedback and advice along the learning session. By collecting information continuously, the environment can make presumptions about strengths and weaknesses of the learner and react appropriately.

Through the theoretical part of the internship, I helped design an agent-based architecture following the TROPOS Methodology ("An Agent Oriented Software Development Methodology"), thus allowing for a deeper understanding of the environment where the agents must operate as well as help designing the key interactions that should occur between different agents (ACL Messages, goals, actions etc). Also, I constructed key sequence diagrams representing important interactions amongst agents for better understanding the kind of ACL Messages that every agent will send and the contents of these messages.

Through the practical part, I choose JADE as a multi-agent development platform to implement the ITS system. JADE (Java Agent DEvelopment Framework) is a software framework to develop agent-based applications in compliance with the FIPA specifications for interoperable intelligent multi-agent systems” [2].

Before we start working in our Intelligent System, I had to understand some basic features such as what are agents, what kind of ACL Messages we can use, what are yellow pages and how does an agent registers/de-registers his services, what is a GUIs platform, as well as more complex features including protocols and contents of ACL Messages (setContent, setPerformative, setConversationId... etc), complex behaviors (CyclicBehaviour, OneShotBehaviour...etc). All these features are well documented in two books that become our main sources of information and that have been a huge help all throughout my internship:

- “Developing Multi-Agent Systems with JADE” [3]

In short, during this internship, my role was to assist in the design of the Multi-Agent architecture and implement the real-time agents essential for the construction of an Intelligent Multi-Agent Tutoring System. Furthermore, I had also been actively implicated in conceiving key tasks regarding database connectivity and user modelling.
3. Project Objectives

The main objective as I have already mentioned is the implementation of real-time Multi-Agent system essential for the construction of the “intelligent” part of the Intelligent Tutoring System. The two sub goal behind the main objective are (1) designing the architecture of Multi-agent system (MAS) followed by (2) implementing our agents inside Multi-Agent platform.

3.1 - Designing the architecture of MAS

For figuring out how to design the architecture of intelligent tutoring Multi-Agent system, I started by studying the classical example of a Multi-Agent System, important for the understanding of agent-based applications with JADE, called the “Buyer and Seller” example. I tried to inquire and analyze agent actions step by step and then design the architecture by using an agent-oriented software engineering methodology called TROPOS. This methodology has a tool that comes as a plug-in in Eclipse called TAOM4E. After mastering this example, I will be required to apply the same principles in building the architecture for the ITS. Below we can see the TROPOS methodology of Buyer-Seller agent, how it looks in TAOM4E environment.

![Figure 2: Example of TROPOS methodology between Buyer-Seller agents, TAOM4E environment](image)

In this figure, we can see the goal dependencies (green) between agents (red). The buyer agent “depends” on seller agent to achieve the goal “sell a book” and seller agent “depends” on buyer agent to achieve a goal “buy a book”. Basically, an agent depends on another agent to accomplish a task that it needs before proceeding with the next series of tasks it has to do.

The main goals that a seller agent must achieve are the following: Firstly, he must register in the yellow pages, the yellow pages service in JADE is in accordance with the FIPA Agent Management specification, is provided by a specialized agent called the DF. Yellow pages service provided by the JADE platform allows any agent to register (publish) services and search for (discover) services
offered by other agents. Registrations, deregistration, modifications and searches can be performed at any time during an agent’s lifetime. The implementation of this example is done in JAVA. We add books with a JPanel class providing general-purpose containers for lightweight components. After we can update the buyer’s catalogue with the inputted book and send a message to the buyer agent to propose offers and prices. When the buyer agents receive a purchase order they process it and remove the requested book from their catalogue.

On the other hand, each buyer agent receives the title of the book that wants to buy. He must search for the list of seller agents in the Yellow pages. When an agent wants to search for services in yellow pages must provide the DF with a template description. The result of the searching procedure is a list of all the descriptions that matches the provide template. If more than one seller agent provides a request to the buyer agent, the buyer agent will only accept the best one (lowest price). Having bought the target book the buyer agent terminates.

After finishing the development of TROPOS methodology using this example, I had the skills to assist in the designing of ITS architecture. Obviously, the buyer and seller agent will be replaced by Tutor and Learner agents as well as other agents having to do with learning.

3.2 - Implementing MAS for an ITS system

After we analyze and understand the goals and dependencies involved in the construction of the architecture of our system, we can continue in the internship by accomplishing the second goal: implement multi-agents and goals in JADE. Firstly, I had to get familiar with the JADE language. Learning how to create an agent, compiling the agent, running it on the platform with specific argument to execute behaviours or terminate the agent. Also, I had to be aware of many issues about agent communication. Get familiar with ACL Messages and the different kinds of ACL Messages that exist. One key aspect I had to find out was how to send, receive and select specific messages with given characteristics from the message queue. Another big chapter to tackle was to Yellow Pages (YP). More specifically, I had to learn how to register, publish and search in the YP for other agents, a very important aspect in building a Multi-agent system.

In addition, I also had to familiarize myself with specific task-related tools. To summaries, I had to become familiar with the platform’s complete, and complex, graphical user interface (GUI). This GUI lets you interact with the agents on your computer, as well as on other computers, using a graphical interface with buttons and symbols, which is easier to use than having to memorize many complicated commands and typing them correctly. The agent platform provides GUI for the remote management, monitoring and controlling of the status of agents, allowing, for example, to stop and restart agents as well as monitoring tools, such as a sniffer agent, to analyze and intercept communications and messages exchanged between agents.
4. **Timeline**

Timelines are particularly useful for they convey not only a sense of change over time but help me achieve goals in a scheduled time-frame. In our timeline we can see the time schedule from the beginning of the internship at the 1st of March 2011 until the 30th of June 2011 and the tasks that we have to complete in every time limit. Since we have some difficulties during the internship, my supervisor and I had to adjust our tasks and objectives accordingly.

<table>
<thead>
<tr>
<th>Week</th>
<th>Dates</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>March 1st to March 4th</td>
<td>Lectures on agents and theoretical constructs</td>
</tr>
<tr>
<td>2</td>
<td>March 8th to March 11th</td>
<td>Lectures on multi-agent systems and initial tests</td>
</tr>
<tr>
<td>3</td>
<td>March 14th to March 18th</td>
<td>Lectures and Java language and JADE platform</td>
</tr>
<tr>
<td>4</td>
<td>March 21st to March 25th</td>
<td>Initial tests with JADE examples, ACL messages, etc.</td>
</tr>
<tr>
<td>5</td>
<td>March 28th to April 1st</td>
<td>Lectures on Multi-Agent methodology and TROPOS</td>
</tr>
<tr>
<td>6</td>
<td>April 4th to April 8th</td>
<td>Implementing the database agent</td>
</tr>
<tr>
<td>7</td>
<td>April 18th to April 22nd</td>
<td>SEPICS oral presentation and initial prototype test</td>
</tr>
<tr>
<td>8</td>
<td>April 25th to April 29th</td>
<td>Implementing the curriculum agent with the rest</td>
</tr>
<tr>
<td>9</td>
<td>May 2nd to May 6th</td>
<td>Lectures on knowledge management and ontologies</td>
</tr>
<tr>
<td>10</td>
<td>May 16th to May 20th</td>
<td>Implementing the Tutor and Learner Agents</td>
</tr>
<tr>
<td>11</td>
<td>May 23rd to May 27th</td>
<td>Implementing the Tutor and Learner Agents</td>
</tr>
<tr>
<td>12</td>
<td>June 13th to June 17th</td>
<td>Integrate changes with the Interface</td>
</tr>
<tr>
<td>13</td>
<td>June 20th to June 24th</td>
<td>Final report and documentation before leaving</td>
</tr>
<tr>
<td>14</td>
<td>June 27th to June 30th</td>
<td>SEPICS Oral presentation and prototype test</td>
</tr>
</tbody>
</table>

Table 1: Timetable of my Internship
5. Work environment

This section will detail the tools and the technologies that we have used in order to begin constructing the ITS. Also, we will try to give a more detailed explanation about the goals and the basic actions of the system. In contrast to previous sections, we will explain some of these tools and technologies in more depth and speak shortly about the action of every module of our ITS system.

This section will cover the following tools and technologies:

- 5.1 - Overall ITS structure
- 5.2 - TROPOS methodology
- 5.3 - GUI platform
- 5.4 - JADE language
- 5.5 - Agent programming
- 5.6 - XML file handling

5.1 - Overall ITS structure

During my internship we achieve to complete the most parts of an Intelligence Tutoring System. Our ITS consists of six different agents:

- The Interface: the relay-point between all agents
- The Tutor: the agent that knows how to teach
- The Expert: the agent that know what to teach (contains a description of the knowledge or behaviours that represent expertise in the subject-matter domain the ITS is teaching)
- The Learner: the agent that represents the learner
- The Database: the agent responsible for connectivity issues with the database
- The Log agent: the agents responsible for recording all interaction with the learner

Interface agent provides the means for the learner to interact with the ITS through a graphical user interface (or other means of communication) and the means for the ITS to interact with the learner by sending interactive feedback. When the Interface Agent receives a message, it sends a message to Tutor agent. In our ITS, the tutor agent must be one of most intelligence agents. A mismatch between a student’s behaviour or knowledge and the expert’s presumed behaviour or knowledge can happen. Subsequently, the tutor takes corrective action, such as providing feedback or remedial instruction. To be able to do this, the ITS needs information about what a human tutor in such situations would do. In our system, these intelligent decisions are given to our system by us in advance. They are not yet dynamic. They consist of a series of “if-else” actions for the moment being. Then the Tutor agent will interact with the Expert agent after agreeing on the next steps to follow. The Expert Agent references to an expert or domain model, which feedback will help Tutor take intelligent and correct decisions. The learner agent uses a learner model, an different learning model for each individual, containing details such as student’s id, knowledge or behaviours, including his score and knowledge gaps. The database agent uses also a database model contains model description (name, location) and the text file (XML document) which exist in the database. When expert agent reply back to interface the specific xml file that interface agent has asked, then interface agent will print one by one every question in the screen for every answer he will take, the
interface agent will send it to Log Agent. Below is the figure of a sequenced diagram of one scenario in the ITS.

In this sequence diagram, we are representing a scenario of a test given by the interface to the learner. This test consists of a series of questions (called steps) that the learner must provide. We can see three things, the agents of our system in the top of the diagram, the sender and the receivers destination of a message pointed by an arrow and the message attributes (inside parenthesis ex. FileName) and values (ex. NextStep) written in the up part of every arrow destination. The reason for designing this sequence diagram was to understand better the sender and the receiver interaction messages so the implementation would be clearer. This diagram represents only one scenario out of the many possible ones that we can have. The ITS developed by my supervisor is a complex system and my internship is primarily concerned with assisting and implementing this specific scenario.

5.2 - TROPOS methodology

TROPOS methodology (An Agent Oriented Software Development Methodology) is a methodology based on goal dependency, a visual modelling tool for object-oriented analysis/design. Before we
decided on using the TROPOS methodology, my supervisor and I read and discussed other methodologies until we agreed that TROPOS would be the most suitable for our needs. One of the alternative methodologies that we checked in detail was entitled “Prometheus methodology”.

After comparing these two methodologies, we preferred to use TROPOS methodology because it provides an early requirements process that goes beyond what is provided by Prometheus. TROPOS gives a detailed design of early, late and architectural requirements, which Prometheus doesn’t. In fact, TROPOS covers the entire life cycle of an agent development project. On the other hand Prometheus provides a more detailed design process but nothing more. This is also a negative aspect to us because that makes TROPOS easier to be learned and used. Furthermore, Prometheus only offers a diagram editor tool that does not go beyond the stage of design.

The diagram of TROPOS methodology was developed in TAOM4E, a tool for working TROPOS methodology in Eclipse. The first diagram which we can see in figure 3 shows us all the goals of the ITS called CHARLIE. We can see that our system has three MAIN goals: model the learner, teach a lesson and record the progress of the learner. Modeling the learner is an important part of any teaching system because it is important to “represent” the learner both in terms of his cognitive as well as emotional state. At the University of Montreal, advanced research is made in Emotional Intelligence in the last years and it is an important part of the current ITS that I will help build.

![Figure 4: Goals of ITS system, in TROPOS methodology](image_url)

The second diagram shows the first goal-scenario of our ITS (Model learner) in more details. We can see all the agents, their actions as well as their dependencies. Below each box we can see the order that each action will be executed. In my internship I will try to accomplish this first goal.
5.3 - GUI platform

Graphical User Interface (GUI) is an agent platform which provides the interface where our agents will live. With a start-up of an agent GUI platform, three main agents are launched. These three agents are included in the GUI platform and are called AMS (Agent Management System), RMA (Remote Monitoring Agent) and DF (Directory Facilitator) agents. Each agent has specific roles and responsibilities. Figure 6 shows the three main agents living in the main container (Main-Container) as well as an interface agent which I have launched from the command line using batch files. Every Agent Platform can have one main Container and many containers in which multiple agents can live and communicate. There is no limit as to the number of agents living in a container.
The AMS agent ensures that each agent in the platform has a unique name and provides supervisory control over access to and use of the Agent Platform. Only one AMS will exist in a single platform. The AMS provides the main functionalities and architectures of the system and the life-cycle services, maintaining a directory of Agent Identifiers (AID) and agent state. JADE agent tables use AID to record agent names and addresses. Each agent must register with an AMS in order to get a valid AID.

The RMA agent is capable for controlling the life-cycle of agent platform and of all active agents inside the platform. The distributed architecture of JADE allows also remote controlling, where the GUI is used to control the execution of agents and their life cycle from a remote host.

The DF agent provides Yellow Pages service by means of which an agent can search for other register agents in YP and provide the services he requires in order to accomplish his goals. By using GUI, the user can interact with the DF by search descriptions of registered agents and modify the description of registered agent. The GUI allows the creation of a complex network of agents.

5.4 - JADE language

JADE language is an agent-oriented programming language. Agent-oriented programming languages are a new class of programming languages that focus on taking into consideration the main characteristics of multi-agent systems. The goal of JADE is to simplify the development of multi-agent systems while ensure the agreement with the FIPA specifications for a complete intelligent multi-agent system. It is important to specify that JADE is built on-top of the JAVA language.

Below, we will give some JADE code examples from our code that they are commonly used in our program. Every agent must do similar actions in the start-up procedure. They must register in Yellow Pages (YP), they must search in YP and in the end of their executions they must deregister from YP. The way to register, deregister and search in YP we can see in the figures below.

```java
DFAgentDescription dfd = new DFAgentDescription();
ServiceDescription sd = new ServiceDescription();
sd.setType("InterfaceAgent");
sd.setName("Agents");
dfd.addServices(sd);
try {
    DFService.register(this, dfd);
}
catch (FIPAException fe) {
    fe.printStackTrace();
}
```

Figure 7: Register in YP
Figure 8: Deregister from YP

Figure 9: Searching the YP for the number of existing agents

Figure 9 presents some very important aspects of agent communication. Indeed, every agent should always look in the YP for the “existence” of another agent required for communication or goal achievement. This is the reason why we use the *DFAgentDescription* class. This class serves as a template to specify the “type” of agent we are interested in searching as well as the description of the provided service. One example would be for a buyer agent to look in the YP using a template of a “book-seller” agent. This type is of course predefined in the application and the seller agent must register using this type. Once we have the different types we can always look in the YP and all every buyer agent will have the most up-to-date seller agent list.

### 5.5 - Agent programming

For the creation of agents we have to define a class extending the jade.core.Agent class and to implement the setup() method. Inside setup method we call a cyclic behaviour method. Below we give an example of JADE code from learner agent followed by a brief explanation.
This code describes a sequence where the Interface agent is asked by the tutor agent to provide the learner model file. The code starts with a private class which is extended as a cyclic behaviour. By writing cyclic behaviour we mean that it will execute in an infinite loop. In line 85 learner agents wait to receive an ACL message. In line 86 we check if message value is different than zero representing the reception of a message. When we receive a message we must check the message’s contents.

Firstly, we check if the getPerformativ() field is equal to a REQUEST message (line 87) and if this is true then we check if the getConversationId() field is equal to the string “Learner-model” (line 88), if this is also true then we have received the correct message and we start execute the code inside the if. Inside the execution, we make the getContent() field equal to Learner id (line 90).

Second, we call a method entitled “RequestPerformer” with two arguments. The step argument which is equal to zero and the learner id that we just import a value (line 92). Inside this new method are all the procedures that agent must accomplish step by step and send back an interactive feedback. If the message is zero then we haven’t receive any message yet (line 97). Every agent works in a similar way. The differences are in the messages that every agent receives and sends and in the goals that they must achieve. The table below shows most of the exchanged messages represented in figure 5 previously described at page 13.

Each start ( * ) next to the step name represents a loop so double starts represent two inner loops. The active learner, meaning the student who will interact with the system is represented with brackets ( {User} ). The content() and conversationID() are ACL message parameters that the agents require in order to properly understand and processes messages between one and another.
Here, the communication protocol of the conversations has already been established and I simply implement it. For example, Instruction_File_REQ in step 8 means that the tutor responds to a request concerning an instruction file made from the Interface agent in step 1.

### 5.6 - XML file handling

XML is an Extensible Markup Language a set of rules for encoding documents. XML was designed to carry data, not to display data. Every agent in our system will have a text file (XML document) for carrying its data. With this way we will be aware on how my data will be stored and can search them and load them easily in memory. More specific, some tasks that I had to accomplish was to read, build, modify and update my data using JDOM (Java Document Object Model) in an XML File in JAVA in the Eclipse environment.

JDOM provides a complete, Java based solution for accessing, manipulating and outputting XML data. An example of how an xml text file, called instructions.xml will looks like is shown in the next figure below. In this xml file we have added three types of instructions that a tutor can ask the ITS to do: give the learner tests, teach him lessons and/or ask him/her questions. In every type of instruction we give details about the name and the address that agent must look for finding the specific xml text file. The reason which we are using instruction file is to be aware of the next steps that our system must perform and thus provide a sense of direction.

---

**Table 2 : Sequence of exchanged messages between agents**

<table>
<thead>
<tr>
<th>Step</th>
<th>Sender</th>
<th>Destination</th>
<th>MESSAGES</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-12</td>
<td>Agents</td>
<td>Agents</td>
<td>Content()</td>
</tr>
<tr>
<td>0</td>
<td>{User}</td>
<td>Interface</td>
<td>LearnerID</td>
</tr>
<tr>
<td>1</td>
<td>Interface</td>
<td>Tutor</td>
<td>LearnerID</td>
</tr>
<tr>
<td>2</td>
<td>Tutor</td>
<td>Learner</td>
<td>LearnerID</td>
</tr>
<tr>
<td>3</td>
<td>Learner</td>
<td>Database</td>
<td>LearnerID</td>
</tr>
<tr>
<td>4</td>
<td>Database</td>
<td>Learner</td>
<td>C:\...\LearnerModel.xml</td>
</tr>
<tr>
<td>5</td>
<td>Learner</td>
<td>Tutor</td>
<td>C:\...\LearnerModel.xml</td>
</tr>
<tr>
<td>6</td>
<td>Tutor</td>
<td>Expert</td>
<td>Next Step</td>
</tr>
<tr>
<td>7</td>
<td>Expert</td>
<td>Tutor</td>
<td>Next Step</td>
</tr>
<tr>
<td>8</td>
<td>Tutor</td>
<td>Interface</td>
<td>C:\...\instructions.xml</td>
</tr>
<tr>
<td>9*</td>
<td>Interface</td>
<td>Expert</td>
<td>Filename</td>
</tr>
<tr>
<td>10*</td>
<td>Expert</td>
<td>Database</td>
<td>Filename</td>
</tr>
<tr>
<td>11*</td>
<td>Database</td>
<td>Expert</td>
<td>C:\...\_FileX.xml</td>
</tr>
<tr>
<td>12*</td>
<td>Expert</td>
<td>Interface</td>
<td>C:\...\_FileX.xml</td>
</tr>
<tr>
<td>13**</td>
<td>Interface</td>
<td>{User}</td>
<td>Ask question</td>
</tr>
<tr>
<td>14**</td>
<td>{User}</td>
<td>Interface</td>
<td>Get the answer</td>
</tr>
<tr>
<td>15**</td>
<td>Interface</td>
<td>Log</td>
<td>Answer</td>
</tr>
<tr>
<td>16**</td>
<td>Interface</td>
<td>Expert</td>
<td>TestID+Array of results</td>
</tr>
<tr>
<td>17**</td>
<td>Expert</td>
<td>Interface</td>
<td>Score</td>
</tr>
<tr>
<td>18**</td>
<td>Interface</td>
<td>Learner</td>
<td>LearnerID+Score</td>
</tr>
</tbody>
</table>

---
To better understand how our agents search for the xml text files and how they request them, I will give a short explanation. Firstly, the Interface Agent will ask the Tutor Agent for the instruction text file (XML document). The idea is for the tutor to decide on the steps to do at a specific time. The Tutor sends the name of the instruction file back to the interface. The Interface will scan the instruction file and for each instruction communicate with the required agent for the specific xml text files. In this case, the 1st test is managed by the expert agent. The expert agent replies back to the interface with the name of the specific xml file that interface agent asked. The interface agent will print one by one every question on the screen and will wait for a reply for each question. For every answer given, the interface agent will send it to the Log Agent. The Log Agent will save every answer in a txt file with some extra details such as a timestamp, learner’s ID, test ID, the specific questions, etc. Also, the Interface Agent will have to collect all answers in a vector. When the test finishes, the interface agent will send this answer vector file to the expert agent for evaluation. The Expert evaluates (computes) the result and sends it back to the Interface. Finally, Interface agent will send sum points to Learner agent for add them in Learner’s Model. It will also send a message to update the Learner’s model in the Database. The Interface agent will go now to the next instruction if there is one.
6. Technical Problems and their Resolutions

During the internship, I faced several technical challenges in the different parts of the objectives I have pursued resulting in valuable time. These difficulties took me of course an important amount of time to complete. In the start of the internship, the difficulties mainly focused on my lack of knowledge in this field of ITS and in the specific terminologies that I had to be aware of before I could start the implementation of theoretical and practical models. Of course I surpassed this problem by studying, reading and getting familiar with the field of ITS as well as the very complicated and exciting aspect of AI (Artificial Intelligence). Most problems arose during the practical and not during the theoretical part of the internship.

In the theoretical part, the main challenge was to be aware of which methodology to use when designing the architecture of the ITS. More specifically, I had to become conscious with the goals, the action and the plans of the desired system so I could find the best methodology that might give us a better representation of our system. A very helpful source for better understanding this was the analysis of the BDI (Belief Desire Intention) model, which is a set of concepts in cognitive psychology for thinking about building Agents that mimic the human thinking process.

In the practical part, I think I would have needed more time to surpass all the technical issues than in the theoretical part. To start with the implementation of the agents, I had to develop good knowledge of the Java language before even programming in JADE. I took some time to program in Java so I had to study a few Java tutorials to advance my knowledge and become capable of achieving future tasks. Also, I had to run some Java examples for getting familiar with the IDE (Integrated Development Environment) in Eclipse which I had never worked in before. I had to understand how to debug, place breakpoints and execute step-by-step programs. After, I become expert enough I could continue with JADE. In order to be accustomed with the JADE language, I had to read a few programming tutorials in JADE for developing my programming skills and analyzing some JADE examples. The most important example is the one that I have already mentioned in a previous section: the buyer and the seller example.

Furthermore, some other technical problem that I had to surpass concerned the XML files. How to manage to read, build, modify and update my data. It was a difficult task because it was my first time working in XML files and I needed some time to understand the structure of an XML document. This information was important to be analyzed for the construction of agent models which will contains the data descriptions of each agent. Some indicative solutions about these technical problems are showed below.
In figure 12 we can see a part of a program which is filtering the data of an XML file and in the second one we can see a part of a program which is building an XML file on disk with specific elements and data.
7. Critical Evaluation

SEPICS program was an excellent chance that was offered to me, from my university in Heraklion on the island of Crete in GREECE, for studying abroad. SEPICS program is a Canado-European exchange program between five European and Canadian Universities (Montreal University, New Brunswick, La Rochelle in France, Glamorgan in Wales/UK and University of Crete in Greece) which have as an ultimate goal, all five university labs to work as one entity in a platform called the Virtual Lab. The first goal of virtual lab is the establishment of a framework for cooperation in higher education, training and youth leading to a curriculum in Intelligent Systems between the European Community and Canada. The second goal is an improvement of the quality of human resources in both the European Community and Canada, by facilitating the acquisition of skills required to meet the challenges of the global knowledge-based economy in the area of Intelligent Systems. The third goal is the development of a new experimental research framework under the concept of virtual laboratory and the last goal is the promotion of the understanding between the peoples of the European Union and Canada including broader knowledge of their languages, cultures and institutions.

In four months period participating in SEPICS program I had the chance to get familiar with AI and specific ITS knowledge. I feel very lucky to have been able to participate in a research project and have the chance to work in a research laboratory recognized worldwide as well as develop my knowledge in new field of Computer Science and enhance my computer skills both theoretically (TROPOS) and practically (JAVA, JADE, XML). Also, I had the opportunity to talk and exchange views with people involved in the field of Computer Science and take opinions that will help me in my future career. Moreover, I had the chance to advance my writing skills and knowledge of English by writing this report and giving two oral presentations for the SEPICS program members.

Besides my experience in my working environment, I also had the chance to experience the life in Canada. I had the opportunity to experience the lifestyle, the habits and contribute in many events and festival which I don’t have the chance to view in my place.

From the day I started working on this project I understood that it will be a great chance for me. They only issue that I have to mention is the problem with the length of the internship. In all the problems that I referred in the previous section, there was always a big issue about time. There were many things to study and get familiar with in a very short period of time. Also, if I consider the time it takes to get used to life here in Canada, especially in the winter that we are not used to, valuable time was lost. So in my opinion I think the SEPICS program would greatly benefit from a time expansion for future internships.
8. References


