Simple Agent Framework: An educational tool introducing the basics of AI programming

M. Pantic, R. Zwitserloot and R.J. Grootjans
Electrical Engineering, Mathematics and Computer Sciences – Mediamatics Department
Delft University of Technology
P.O. Box 5031, 2600 GA Delft, the Netherlands
mpantic@ieee.org, reinier@zwitserloot.com, robbertjan@dds.nl

***Abstract*—This paper describes a Java-implemented agent framework developed for the purposes of an introductory undergraduate course of Knowledge Engineering. Although numerous agent frameworks have been proposed in the vast body of literature, none of these available agent frameworks is simple enough for the usage by first year undergraduate students of computer science. Hence, we set out to create our own framework that would fulfill all requirements, satisfying the aims of the course, the computing skills level of the intended group of students, and the size of the intended group of students. Besides the designed agent framework, which embodies the concepts of concurrency, multi-agent systems, and persistency, the strategy decided upon the best possible utilization of the developed tool for the goals of guiding and instructing the students in their learning of AI concepts and techniques, is also discussed. The results of the coursework suggest that the developed agent framework is highly suitable for the purposes of teaching the students the AI basics including the knowledge representation schemes, rule-based reasoning and intelligent agents paradigm.**

***Index Terms*— educational tools, intelligent agents, agent framework, rule-based systems, semantic networks, Java**

I. Introduction

According to the Dutch Statistical Bureau, over 69% of Dutch households use a PC [1]. This ever-increasing role of computers in our society clearly forecasts the type of working environments and information-communication spaces we are about to use in our everyday activities. Namely, even nowadays the majority of people in our country exploits computers for work and uses the Internet to communicate with each other, to shop, to seek out new information, and to entertain themselves. This clearly indicates that in the future, with the aid of computers, we will carry out our daily tasks, we'll communicate and entertain ourselves in cyberspace across distance, cultures and time. Of course, the specifics of such cyber worlds and smart environments and of pertinent interfaces, which should facilitate easy and natural communication within those environments and with the variety of embedded computing devices, are far from settled. Hence, computing technology breakthroughs are compulsory. This necessity forms the main drive behind the abundance of job offers for IT specialists in the Netherlands. It also explains why so many of our prospective young minds choose just this branch of exact sciences to examine their capabilities, to enhance their skills and to develop themselves into valuable experts in one of the IT fields. At the very least, as computers become ever more ubiquitous in our society, a further development of computing technology becomes one of the most exiting and economically important topics for both the professional and scientific sector.

It is this view on reality that has motivated the development of a new educational program, called Media and Knowledge Technology, of the Computer Science at Delft University of Technology, the Netherlands. In the academic year 2001-2002 this new program has been officially introduced. The main objective of this program is to educate students to become engineers who are able to design and develop intelligent systems for multimedia and multimodal information and knowledge processing and who are able to design, realize, and deploy properly working man-machine interfaces.

As a part of this program, an introductory first-year undergraduate course on Knowledge Engineering has been introduced with the main aim of achieving two different but overlapping goals:

1) to introduce the basic concepts of knowledge engineering and relevant artificial intelligence (AI) techniques, and
2) to explain and instruct on issues related to AI programming in general and intelligent (multi-) agent applications in particular.

In contrast to the classic notion of AI, which represented a promise of intelligent machines with abilities comparable or possibly superior to those in humans [2], this course has been envisioned to approach AI as a set of techniques for making software that is more intuitive and easier to use and which makes users more productive (e.g., as proposed in [3]). The AI techniques handled in this course include search algorithms, knowledge acquisition and representation techniques, rule-based reasoning algorithms, and distributive AI techniques focusing on agent technology. The second part of this course has been envisioned as to pertain to various practical issues in creating intelligent agents and understanding intelligent agent applications. It has been envisioned to build on the first part of

The work of Maja Pantic is supported by the Netherlands Organization for Scientific Research (NWO) Grant EW-639.021.202.

the course by taking AI algorithms and using them for the development of intelligent agent applications aimed at monitoring, filtering and retrieval of relevant information from Internet and Web pages. The overall course has been envisioned to include 4 hours of weekly lectures (part one of the course) and 40 hours of practical work (part two of the course) during the second half of the second semester.

II. Teaching Material Requirements

Several requirements have been imposed on the selection of the appropriate teaching material to be used in the second part of the introductory course in question. Given that the students attending the pertinent course are the first year students who have been previously thought only the Java programming language, all programs and examples to be learned and developed in the course have to be Java-coded. Another important issue, valid for any introductory course, concerns the focus of the envisioned practical coursework – it should be on learning how to implement the principle AI techniques listed above rather than on learning complex software to be used as a tool for supporting the learning process. In summary, the requirements that are to be fulfilled in order to come up with an appropriate selection of the teaching material to be used have been defined as:

1) The tool to be used by students should support the development of intelligent agent applications aimed at monitoring, filtering and retrieval of relevant information from Internet and Web pages.
2) It should be Java-based, easy to use, and facilitating the students to use built-in Java-implemented agent "templates", to edit them, and to include the desired AI algorithms according to the goals of the current exercise.
3) It should embody the concepts of concurrency (a kind of multi-threaded setup), multi-agency (by allowing simple communication from one agent to the other), and persistency (saving settings between executions).

Although numerous agent frameworks have been proposed in the vast body of literature (e.g., [3], [4], [5], etc.; see [6] for a very large database of the related works) and although there are now several commercially available software packages enabling the development of agent-based applications (e.g., CIAgent [3], ADE [7], ADK [8], AgentSheets [9], etc.; see [10] for an almost exhaustive list of the existing commercially available tools), none of these available agent frameworks satisfy all the requirements delimited above. Some are not simple enough for the usage by first year undergraduate students (e.g., CIAgent [3], ADE [7], ADK [8]). Others could prove suitable in regards to complexity (i.e., taking into account the computing skill level of the intended group of students), but missed other important properties. For example, AgentSheets [9] allow a user with no programming experience to develop agent-based applications in Java but the pertinent software is available only for Mac workstations, which we do not have at our university labs. Hence, we set out to create our own framework that would fulfill all the requirements delimited above and yield an appropriate tool to be used in the practical coursework in question.

III. Simple Agent Framework

The first step in any software development project is the collection of requirements from the intended user community. In our case, this was made difficult because the students considering to attend the newly introduced first-year course on Knowledge Engineering could not provide this kind of feedback until we had already designed, developed, and deployed the product (the *Simple Agent Framework* tool used in the practical coursework). Yet, as already explained above, we made some obvious decisions based upon our knowledge about the intended users and the stipulated purpose of the educational tool in question. We decided to develop a platform supporting the development of intelligent agents using Java. We also decided to provide the ability to add intelligence to built-in agent templates written in Java. This also meant that we were supposed to implement (partially) the basic artificial intelligence techniques listed in the introduction and to enable the re-usage of this Java code. Focus on the topic at hand – intelligent agents – was another requirement. Yet, we would be doing the students a disservice if we spent large amounts of time developing communications code, an object-oriented DB, or a mechanism for performing remote procedure calls that are all out of the scope of the educational goals of the course in question. In summary, we wanted to develop a Java-based framework with a simple yet flexible architecture that is focused on intelligent agent issues.

Intelligent agents can be viewed either as adding value to a single standalone application or as a freestanding community of agents able to interact with each other and other applications [4]. The first is an application-centric view of agents, where the agents are helpers / strollers in the application. The second is a more agent-centric view of agents, where the agents are able to monitor and drive the applications. The agent framework that we intended to develop was envisioned as being easy to understand and straightforward to use. The primary aim of this framework was (and still is) to illustrate how intelligent agents and different AI techniques can be built and combined to yield intelligent applications. Hence, we chose application-centric approach as the basis for the architecture of our intelligent agent framework. This approach is the least complex because agents can be regarded as simple extensions of the application functionality yielding a so-called Nouvelle-Expert-System-oriented application [11].

Simple Agent Framework can be seen as a common programming interface delimiting the behavior of all the agents integrated into the framework. Its functional specifications can be summarized as follows:

- *Simple Agent Framework enables an easy addition of intelligent agents*. We could achieve this goal by facilitating the framework to instantiate and configure the agent and then call the agent's methods as service routines. That way the framework would be always in control, and it could use intelligent functions as appropriate. This is easy, but this is hardly what we would consider an intelligent agent. Another possibility was to have the framework instantiate and configure the agent and then start it up in a separate thread. This would give the agent some autonomy, although it would be running in the framework's process space. This is the design we chose.

- *Simple Agent Framework supports a simple event processing, allowing the agents to handle the events coming from the outside world or from other agents and to signal events to the outside world*. Java uses an event-processing model for various features, including the features of the graphical toolkit (swing). However, this model relies on the source of events to maintain a list of registered event listeners and to deliver the subject events. To alleviate the difficult task of programming agents, a new event processing system has been designed for the Simple Agent Framework. It is called a "delivery system" and it manages and dispatches events to the interested agents. Both the events coming from the user and the events coming from other agents are represented as text strings, each of which is called a *message*. All messages are sent to a *channel*, each of which is analogous to a blackboard. The blackboard concept is known to our first year students since it is the design pattern used to develop the system for posting and distributing information about courses, exams, homework, etc. To be able to receive messages, an agent should register for a channel. Once registered, it would be notified of messages delivered to that channel. Messages are received through a locally-defined (i.e., at the agent level) *handle(channel, message)* method. To send a message, an agent should invoke a globally-defined (i.e., at the framework level) *write(channel, message)* method. A complete list of methods supported by the framework is given in Table 1.
- *Simple Agent Framework supports adding domain knowledge and intelligence to agents.* Facilitating a composite design has attained this functionality of the framework. Namely, we designed and developed forward and backward rule-based inference procedures, rule-base constructs, semantic networks constructs, and several search algorithms in Java. The student can use the related Java classes to provide the pertinent functions to his/her agents.
- *Simple Agent Framework supports the concept of concurrency needed to allow agents to operate independently and yet at the same time*. This has been achieved by starting each agent in a separate thread, allowing it to access the delivery system described above at its own convenience. To prevent overloads, the framework initiates a queue of messages for each agent, stores it locally (i.e., at the agent level), and processes it in a one-at-the-time fashion.
- *Simple Agent Framework supports the concept of persistence*. Enabling agents to instruct the framework to store values referenced by a key has attained this. The framework retains this *(key, value)* pair and allows access to it at any time, even when the execution of the framework has been ceased in the meantime. To do so, the framework saves *(key, value)* pairs to a text file and loads them in when its execution is commenced once again.
- *Simple Agent Framework is a graphical agent-building tool*. We wanted a direct-manipulation interface in which WYSIWYG (what you see is what you get) would be the guiding principle. Yet we wanted a simple and comprehensible GUI that could account for differences in computing skills and experience of the intended users. Given that the target users were first year students with rather limited computing skills in the majority of cases, we wanted to omit extensive technical terminology, irreversible and user-uncontrolled unconcealed actions, complex screen layouts, incomprehensible error massages and unexpected crashes. To attain this, we developed a rather simple GUI which is easy to grasp, use and create. The students' feedback provided at the end of the course affirms this. This feedback also gained us an insight into how to improve GUI in a next version of the tool (e.g., including help files about the framework, a tutorial on Java, direct links to exercise descriptions, etc.). The developed GUI has three main screens listing active *Agents*, available *Channels*, and existing *Properties*. A double-click on a property (i.e., a *(key, value)* pair) allows the user to change it. A double-click on a channel opens a *channel viewer*, which displays messages in the order in which they were received and allows the user to insert a new message manually. The later proved to be very helpful for testing implemented agents. The agent screen features options such as load in an agent and shut down an agent.

The Simple Agent Framework uses two primary classes. The *Manager* is the base class that defines a common programming interface and behavior for all the agents in the Simple Agent Framework. Within this class, all four global methods listed in Table 1 are defined. All the agents in the Simple Agent Framework communicate with the environment and each other exclusively by using these methods. Each agent extends the abstract *Agent* class and uses modified versions of *handle* and *init* methods given in Table 1. Consequently, users of the Simple Agent Framework need only to know about these two classes in order to be able to create new, Simple Agent Framework compatible agents.

TABLE I. THE METHODS SUPPORTED BY THE SIMPLE AGENT FRAMEWORK

Framework-level methods	***Agent-level methods***
setProperty(*key, value, persistent*) This method allows the agent to change an existing or to create a new *(key, value)* pair. A flag *persistent* facilitates the agent to define whether the framework should store the property between sessions.	**Init()** This method is called, in a separate thread, when the agent is loaded.
getProperty(*key*): *value* This method returns values set by using the *setProperty* method.	
register(*channelname*) This method registers the agent for a channel. If a message has been sent to the pertinent channel, the agent's *handle* method is called.	**handle(*channelname, messagecontent*)** Messages that have been sent to a channel for which the agent has been registered are received and handled by this method.
write(*channelname, messagecontent*) This method allows the agent to send messages to any channel.	

IV. Coursework

Two main exercises constitute the practical coursework. The first one delves into the issues of how to incorporate a rule-based reasoning into an intelligent agent and then use it for constructive purposes such as to rank available information according to a set of stipulated preferences. The 2nd exercise focuses on constructing intelligent agents using the semantic network concepts and deploying them to monitor, filter, and retrieve relevant information from Internet and Web pages. In both cases, the subject problems have been defined so that the students are incited to program a variety of Simple Agents, which in collaboration achieve the goal of the exercise.

In order to alleviate "hard-core" programming tasks and, in turn, to free students' time for efforts in understanding and exploring AI concepts and applications, chunks of codes have been handed out. Namely, a couple of example agents have been included into the Simple Agent Framework based upon which the students could commence building their own agents (e.g., by editing, enlarging or enhancing the existing code). This also enabled the students to explore the framework and different agents while "in action", prior attempting an actual design of agents. As already noted above, we developed rule-based inference procedures, rule-base constructs, semantic networks constructs, and several search algorithms in Java. The related Java classes were made available to the students.

The fostering of teamwork skills and spirit is considered to be an objective of immense importance in the educational programs of Delft University of Technology. Hence, virtually all the existing practical courses, including the subject course, have been designed for teams of students. In our case, 5 or 6 students constituted a team.

A. *The First Exercise: Rule-Based Reasoning*

The first exercise has been defined as follows.

Create an agent-based system that translates a questionnaire, filled in by each participant in this course, into a chart that expounds the suitability of each participant for being a part of your team. Use Simple Agent Framework to build the required system and employ rule-based reasoning. Explain the choice of the utilized inference procedure and the final ranking of the students being a part of your team.

Each of the 14 teams that participated in the course, approached the posed problem in the following way (Fig. 1):

1) Build a *Reader Agent* that retrieves for each student who filled in the questionnaire his/her answers from the *Survey Data* Web page.

2) Build an *Inference Agent* that applies forward chaining inference procedure to determine the profiles of the fellow students based upon their "survey data". To do so, develop a knowledge base containing a set of rules for delimiting the profile of a student. The utilized questionnaire has been designed in a way that allows students to define a large variety of such rules. For example, based upon an affirmative answer to the question "Do you often take the initiative in a project team?", different teams defined different rules for assigning labels such as *leader*, *arrogant*, and *stupid*. By combining different "survey data" and scored labels, the *Inference Agents* developed by different teams generated values like *creative*, *dependable*, *nerd*, etc.

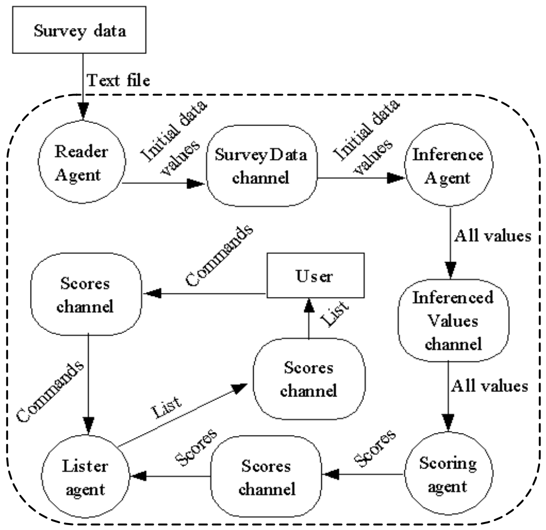

Fig 1: General structure of the multi-agent system to be developed in the first excercise

3) Build a *Score Agent* that assigns to each of the fellow students an estimate of his/her suitability for being a part of the team. For example, being *dependable* can be decided to be a desired property of a student, and given a score of +10, while *all knowing* can be deemed to be an unwanted aspect, and given a score of -5.

4) Build a *Lister Agent* that assembles a ranking list by sorting the scores provided by the *Score Agent*.

5) Provide the required explanations. Estimating whether the students understood the difference between forward and backward chaining formed the main incentive for asking the students to explain their choice of the inference procedure. Inciting students to think about their roles in the team motivated the request to explain the final ranking generated by their system. For example, if a member of a particular team was estimated to be unsuitable for the pertinent team, then either the subject member did not participate in the development of the utilized rules and *Score Agent* or he was not able to defend his standpoints on how the mechanisms in question are to be developed.

B. *The Second Exercise: Semantic Networks*

The second exercise has been defined as follows.

Create an agent-based system that retrieves and analyses BBC news available via Internet according to your preferences. Use Simple Agent Framework to build the required system and employ the semantic network concepts. Explain the drawbacks (if any) of the utilized approach.

Each of the 13 teams that accomplished the subject task, approached the posed problem in the following way (Fig. 2):

1) Build a *Reader Agent* that monitors the BBC Web site and flags the system when a novel news article is posted.

2) Build a *Make Network Agent* that constructs a semantic network representation for each article. To do so, label each word as either *trivial* (words like a, an, the, is, has, etc.) or *non-trivial* and represent each non-trivial word as a node of the network. Associate a *relevance* property with each node to expound the frequency with which the related word occurs in the current article. Connect the nodes of the network and assign a *match* property to each such link to expound the frequency with which the subject connection between words occurs within a single sentence. To aid the students with debugging, we provided an agent capable of printing out an entire semantic net.

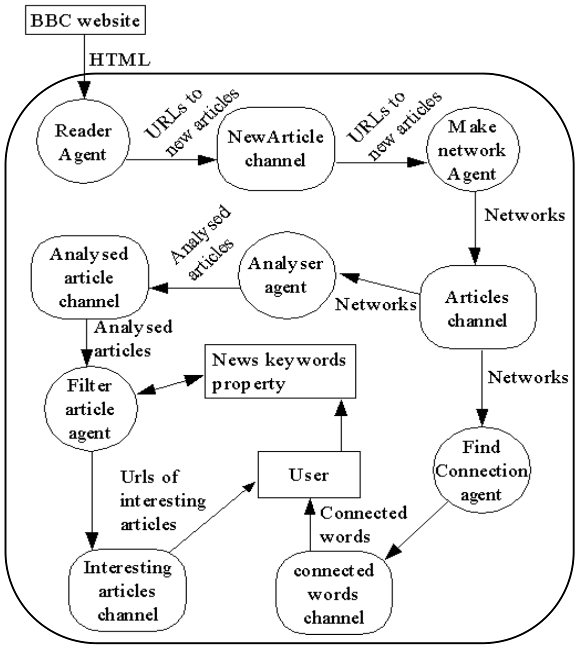

Fig 2: General structure of the multi-agent system to be developed in the second excercise

3) Build an *Analyzer Agent* that selects ±5 nodes of the semantic network having the highest values associated with their *relevance* property and outputs the selected nodes, their links to other nodes, and the pertinent linked nodes themselves.

4) Build a *Filter Article Agent* that labels the examined article as being either important or unimportant. To do so, check whether the word related to a selected node is in the list of keywords (i.e., in the list of preferred topics). Check also whether a selected node has a strong link with another node (use the value assigned to the *match* property of the links) and, if so, check whether the word related to that node is in the list of keywords. If either of these checks is in the affirmative, label the analyzed article as being important. As an example of both a basic machine learning technique and a task that an agent-based system could do for us, we provided the students with the *Find Connection Agent* which monitors the generated semantic networks and outputs the series of usually connected words (e.g., "World Trade Center", "Robert Mugabe", "Tony Blair", etc.).

5) Provide the required explanation. Inciting students to think about the fact that homonyms and synonyms might (and probably will) affect the results generated by their system formed an incentive for asking the students to list the drawbacks of their system. Another incentive for doing so was to make an estimation of whether the students understood the drawbacks of a utilized search algorithm, the advantages that could be achieved by including a "trained" *Find Connection Agent* into the filtering process, etc.

V. Experiences and Conclusions

The evaluation of the utilized educational methods and materials by students is considered to be an objective of immense importance in the educational programs of Delft University of Technology. Based upon the feedback provided by students, the courses and the utilized tools and readings could be improved to fit better current students' knowledge and skills, their preferences, and overall and/or specific goals of the entire subject educational program. There is a standard questionnaire available for eliciting students' opinions on computer science courses, which includes all the relevant questions about the suitability of the used tools and readings, the experiences of the student during the course, and the ways the course could be enhanced. From 73 students who attended the course on Knowledge Engineering, 68 filled in the pertinent questionnaire. In 67% of cases, students evaluated the utilized educational material as being suitable for the goals of the course. In 78% of cases, students labeled the practical coursework as being interesting and motivating. In 83% of cases, the students claimed that they enhanced their skills and acquired new, valuable knowledge on the instructed subjects. These results suggest that the developed Simple Agent Framework and the designed exercises are highly suitable for the purposes of teaching the students the AI basics including knowledge representation schemes, rule-based reasoning and intelligent agents paradigm. They also suggest that the students liked the coursework and regarded it as motivating.

Nevertheless, the students indicated several aspects of the Simple Agent Framework that could be enhanced in a next version of the tool. One pertains to the enhancement of the tool's GUI. As already noted above, students' suggestions on this issue concern the inclusion of more elaborate help files, a tutorial on Java, and direct links to exercise descriptions.

Currently, if an agent enters an infinite loop, other agents and the interface become unresponsive and the framework must forcibly be shut down. This is caused mainly due to the threaded nature of the Simple Agent Framework. Nevertheless, given that the students may (and usually do) make programming mistakes, it is necessary to make the framework more robust. For instance, eliminating automatically the agents that are using too many system resources could attain this.

Addressing the two challenges explained up to this point and, in turn, developing a better version of the Simple Agent Framework, is a logical and necessary extension of our current work. In addition, efforts towards enabling the agents of the Simple Agent Framework being mobile could open up a fruitful avenue for introducing students to both the concept of mobile agent and the potential usefulness of such agents.

References

[1] http://www.cbs.nl/nl/publicaties/artikelen/algemeen/webmagazine/artikelen/2001/0808k.htm.

[2] J. Haugeland, *Artificial Intelligence: The Very Idea*. Cambridge, MA: MIT Press, 1985.

[3] J.P. Bigus and J. Bigus, *Constructing intelligent agents using java*. New York: John Wiley & Sins, 2001.

[4] K.A. Arisha, F. Ozcan, R. Ross, V.S. Subrahmanian, T. Eiter and S. Kraus, "Impact: A platform for collaborating agents", *IEEE Inteligent Systems*, vol. 14, no. 2, pp. 64-72, 1999.

[5] G. Weiss, Ed., *Multiagent Systems*. Cambridge, MA: MIT Press, 1999.

[6] http://www.agentlink.org/press/

[7] http://samuel.cs.uni-potsdam.de/soft/taxt/research/ade/ade.html

[8] http://www.tryllian.com/index.html

[9] http://agentsheets.com/

[10] http://www.agentlink.org/resources/agent-software.php

[11] Y. Shoham, "What we talk about when we talk about software agents", *IEEE Inteligent Systems*, vol. 14, no. 2, pp. 28-31, 1999.