Teaching Ad-hoc Networks Using a Simple Agent Framework

Maja Pantic, Reinier Zwitserloot, Robbert-Jan Grootjans
Delft University of Technology
Electrical Engineering, Mathematics and Computer Science
Mekelweg 4, 2628 CD Delft, The Netherlands
mpantic@ieee.org, surial@gmail.com, grootjans@gmail.com

Abstract - **This paper describes a method of teaching Agent Technology and Ad-hoc Networks using a novel, simple agent framework developed specifically for the purposes of teaching introductory Artificial Intelligence (AI) to undergraduate students. The agent framework in question is Java-implemented and it embodies the concepts of concurrency, multi-agency, persistency, and mobility. The introductory AI coursework in question is a set of assignments that requires the students to use intelligent agents to route SMS messages through an ad-hoc network. While many AI courses teach basic AI concepts such as heuristic search, rule-based reasoning, and neural networks, by means of programming assignments, we were not able to find any AI course introducing students to relatively new AI concepts such as distribution, mobility and ad-hoc networks by similar means. The coursework presented in this paper represents a synthesis of the traditional objectivist approach and a real-world oriented, constructivist approach to introducing students to these "hot topics".**

Index Terms – Agent Framework, Ad-hoc Networks, Mobility, Programming Assignments.

INTRODUCTION

Short range communications capabilities such as Bluetooth [1] and WiFi [2] are becoming increasingly present in small mobile devices, such as cell-phones and PDA devices. The availability of this technology will become increasingly ubiquitous just as most mobile devices have SMS capabilities nowadays. It is inevitable that the field of ad-hoc networks will play an important role in the development of this kind of novel technologies.

The hype about multi-agent systems (MAS) is another fact. Since 1995, many of the IEEE and ACM journals have devoted special issues to software agents. Since 1997, many agents conferences have taken place. Nowadays, special panels on agents are organized during industry symposia. Agents that sort emails [7], retrieve and parse information from Web pages [8], and represent personal assistants having a 'personality' [9] are commercially available to date.

Nonetheless, while many AI courses teach basic AI concepts such as heuristic search, rule-based reasoning, and neural networks, by means of programming assignments, we were not able to find any AI course introducing students to these relatively new AI concepts, concerning distribution, mobility and ad-hoc networks, by similar means. The coursework presented in this paper represents a synthesis of the traditional objectivist approach and a real-world oriented, constructivist approach to introducing students to these "hot topics". More specifically, the coursework builds on the theory of AI by teaching students how to use MAS and ad-hoc network concepts for solving a well-defined assignment (objectivist approach) aimed at sending Short-Message-Service messages across an ad-hoc network (constructivist approach).

This paper will begin by providing the background information about the introductory AI course to which the coursework described in this paper belongs. Then, the simple agent framework on which the pertinent coursework was built will be explained. The assignment forming the coursework in question will be described next. Finally, classroom experience will be discussed.

INTRODUCTORY COURSE ON AI

The coursework described in this paper is a part of an introductory first-year undergraduate course on AI which was created as a part of a new educational program, called Media and Knowledge Technology, for the Computer Science studies at Delft University of Technology, the Netherlands. This course was introduced in the academic year 2001-2002. The main aim of this course is to achieve the following:

- Introduce the basic concepts of knowledge engineering and relevant AI techniques including search algorithms, knowledge representation methods, rule-based reasoning algorithms, and agent technology.
- Explain and instruct on issues related to AI programming in general and intelligent MAS in particular.

The classic approach to teaching AI is an objectivist approach [6]. In contrast, this course departed from the orthodox objectivist approach to teaching programming to novices where assessment exercises have no real connection to how the student will apply the newly obtained knowledge and skills to previously unseen, real-world problems [3]. An alternative, constructivist approach, where an authentic real-world environment is provided in which students apply and test their newly obtained knowledge and skills, seemed to represent a better choice for the AI course in question. Recent studies suggested, however, that the first-year undergraduates are not ready for a pure constructivist teaching approach, in which they are given a vaguely specified problem statement that they should refine and for which they should then develop a solution [4]. Hence, a pragmatic synergy of objectivist and constructivist approaches to teaching programming to novices has been adopted for the AI course in question [3].

The course has been envisioned to include 20 hours of lectures (part one of the course) and 80 hours of practical work (part two of the course). The practical work was designed to build on the first part of the course by teaching students how to create intelligent MAS using AI techniques about which they have been lectured. In 2002 the practical part of the course consisted of two assignments [5], [6]: one focusing on rule-based reasoning and the other on semantic networks. In 2003 two additional assignments were introduced [3]: one focusing on epistemic logic and the other on the concept of mobility. As students found the latter to be too complex and abstract [3], it was replaced by the assignment described in this paper.

FLEEBLE AGENT FRAMEWORK

The students attending the AI course in question are first-year undergraduate students who have only attended a course on Java programming language. This and the fact that the focus should not be on learning how to use complex software tools, imposed a number of requirements that an educational tool should fulfill in order to be appropriate for the purposes of the AI course in question. These are:

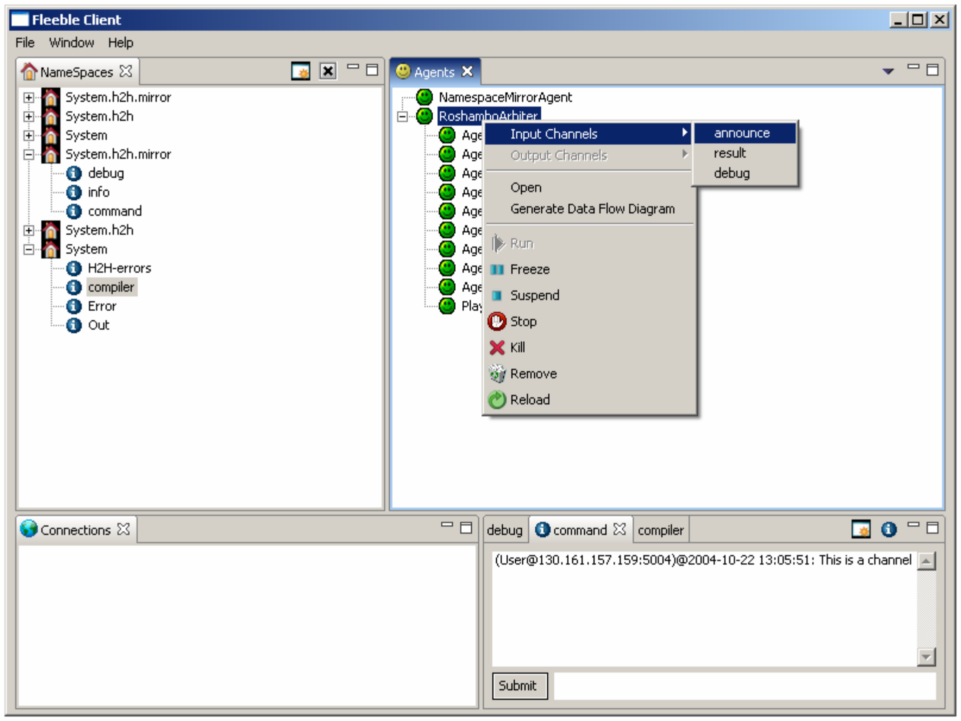
- The tool to be used should support the development of intelligent (and Internet-oriented) agent applications.
- It should be Java-based and easy to use. It should facilitate the usage of built-in Java-implemented agent "templates" that can be easily extended to include the desired AI algorithms according to the goals of the current assignment.
- It should embody the concepts of concurrency (a kind of multi-threaded set-up), multi-agency (by allowing simple communication from one agent to the other), persistency (saving settings between executions), and mobility (by allowing exchange of messages between agents residing on different frameworks).

In 2001, when the AI course in question was introduced for the first time, numerous software packages enabling the development of agent-based applications were available [6]. Nevertheless, none of the available agent frameworks satisfied all the requirements listed above. The reason is that virtually all of these frameworks had at least one of the two drawbacks: they lacked readable documentation and readily available examples and/or they lacked a self-explained easy-to-use GUI. Hence, the authors set out to create a novel Java-based agent framework that would fulfill all the requirements listed above.

In 2002, we developed a first prototype of the framework, which we called Simple Agent Framework (SAF) [5]. The aspects of SAF in need of enhancement, as indicated by the students who evaluated the course material, concerned the improvement of the GUI, enabling agents to load/instantiate other agents, and solving problems caused by the threaded nature of SAF [5], [6]. The realization of these aspects led to a novel version of SAF [6]. This tool was used for the 2003 edition of the course. Finally, we wanted to include a novel assignment in the coursework with the goal of introducing students to the concepts of distribution, mobility, and ad-hoc networks, the concepts that make the agent technology such a hot topic. The realization of these aspects led to a new design of SAF, which we named *Fleeble*.

Fleeble can be seen as a common programming interface delimiting the behavior of all the agents integrated into the framework. Its functional specification can be summarized as follows [3].

- *Fleeble enables easy addition of intelligent agents*. The design that was chosen is to have the framework instantiate and configure the agent and then to start it up in a separate thread [6]. This gives the agent some autonomy, although the agent is running in the framework's process space. An agent can also instruct the framework to start up another agent. The framework keeps track of all agents and their parent agents. So, a single agent can be created which starts up the appropriate agents for each multi-agent system. This *kickoff* agent is then the parent agent of all agents that form the multi-agent system in question. Hence, loading one or more multi-agent systems will result in one or more easily traversable trees that will clearly indicate which agents are working together to accomplish a single task (Fig. 1).
- *Fleeble supports simple event processing, allowing the agents to handle events coming from the outside world and other agents and to signal events to the outside world*. Fleeble offers a message distribution system for communication between agents that is based on a Publish/Subscribe system, which is centered on the concept of a *channel*. Channels are named entities that allow a single message to be delivered to any number of agents, using the following mechanism. An agent informs Fleeble that it is interested in events pertaining to a specific channel (i.e., it subscribes to that channel). An agent can ask Fleeble to deliver a message to a channel (i.e. it publishes to the channel in question). Fleeble creates a "handler" thread for each agent that has subscribed to the channel in question. All handler threads are started simultaneously and deliver the message to the subscribed agents.
- *Fleeble supports adding domain knowledge and intelligence to agents*. A number of AI algorithms were designed and developed in Java, including forward and backward rule-based inference procedures, rule-base and semantic networks constructs, and several search algorithms. The students can use the related Java classes to provide their agents with these functionalities [6].



```
(User@130.161.157.159:5004)@2004-10-22 13:05:51: This is a channel
```

FIGURE 1: GUI OF FLEEBLE AGENT FRAMEWORK

- *Fleeble 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 [6].
- *Fleeble supports the concept of data persistence*. This has been achieved by enabling Fleeble agents to instruct the framework to store values referenced by a key. The framework stores this (key, value) pair and allows access to it at any time, even when the execution of the framework has ceased in the meantime [6].
- *Fleeble supports the concept of state persistence*. State persistency allows the programmer to shut down a single agent (or even the entire framework) and to restore it from the point where it was suspended later on, even when the computer has been shut down in the meantime. A Java-implemented state persistency forces certain restrictions onto the programmer, such as the requirement to return control to the framework within a reasonable time span even if the agent was not stored since it was in the middle of processing an instruction. Allowing the agent (that is to be stored) to complete the processing of the current instruction minimizes this problem. Once the current instruction has been handled, the agent will be stopped, and Java serializing will be used to save all pertinent objects (the agent state) to the permanent storage.
- *Fleeble enables the user to handle malfunctioning agents*. The main drawback of the used threaded approach becomes apparent when a malfunctioning agent ends up in a loop and crashes the whole system, causing other, properly functioning agents to become unresponsive as well. As a solution, the GUI of Fleeble runs in a different Java virtual machine and the communication thread is given the highest priority which ensures the GUI will be responsive to the user at all times. Since any item of an agent tree can be suspended, reawakened, or even destroyed, the user is enabled to manually find and destroy any malfunctioning agent.
- *Fleeble supports Object-based Communication***.** In the first versions of the framework this capability was limited to sending only simple text strings [5], [6]. In the current version of Fleeble any object can be sent via the delivery system. The ability of Java to inspect objects on the attribute level (using the reflection API) is used to display the contents of a sent object to the user. This removes the need to program the conversion of a given object into a string and back again, which was necessary in the previous versions of the framework [6].
- *Fleeble supports the concepts of Distribution and Mobility*. Distribution is the possibility to exchange messages between agents residing on different frameworks. It has been enabled by establishing socket-to-socket connections between frameworks. Fleeble manages these connections; it creates and closes them as needed. Connections can be used for mirroring channels, translating the concept of a channel into something that transcends the host machine of the agent. Connections are also used to transport agents to another host (see the State Persistency explained above), allowing agents to physically move. This process is called agent mobility. The process of being moved to another host can be started either by the agent or by its parent agent.
- *Fleeble enables the programmer to emulate multiple frameworks***.** When testing an environment designed for distributed multi-agent systems, it is not always feasible to reserve a number of computers needed for the pertinent testing. To support proper testing of agents designed to function in a multiple-host environment, Fleeble offers the ability to encapsulate all channel communication for any agent such that the agent “gets the impression” that it

has moved to a different host. Agents can communicate with each other (i.e., use the same channels) as long as they run under the same virtual framework.

- *Fleeble supports auto-compilation***.** During the software development process, code is usually improved incrementally. This makes it necessary to compile the code frequently. Doing this with an external compiler is time-consuming. To alleviate this problem, we adapted Fleeble so that it detects changes in the source code and automatically recompiles all files that have been changed, even if a multi-agent application is still running. A single click on the *reload* button starts the process of recompiling the source and replacing the running agents with the new version of these agents.
- *Fleeble has a simple, easy to understand GUI.* The goal was to develop a direct-manipulation GUI in which WYSIWYG (what you see is what you get) would be the guiding principle [6]. A simple self-explicatory GUI was developed that is easy to understand and use. As can be seen in Fig. 1, the GUI of Fleeble visualizes the overall traffic through channels, the agent hierarchy, and the agents themselves. By clicking the right mouse button on an appropriate entry, the user opens the relevant context menu with the related actions. This menu enables quick perception of and access to the relevant data.
- *The Fleeble GUI supports monitoring of external connections***.** Fleeble makes an active network connection when it should transfer a message or mobilize an agent to a remote platform. Connections to remote platforms and the related information such as the address and the port number are displayed in the connection view (see Fig. 1).
- *The Fleeble GUI supports remote connections to a Fleeble server***.** The Fleeble GUI runs independently of the main agent framework and it does not need to be started in the same physical computing environment in which the agent framework runs. A user can log into a Fleeble server either locally or remotely, via a network connection. The amount of control a remote connection can assert over the agent framework is nevertheless rather limited. Operations such as loading an agent are not possible from a remote location.

ASSIGNMENT ON AD-HOC NETWORKING

The programming assignment teaching concepts of mobility, distribution, and ad-hoc networking, is as follows:

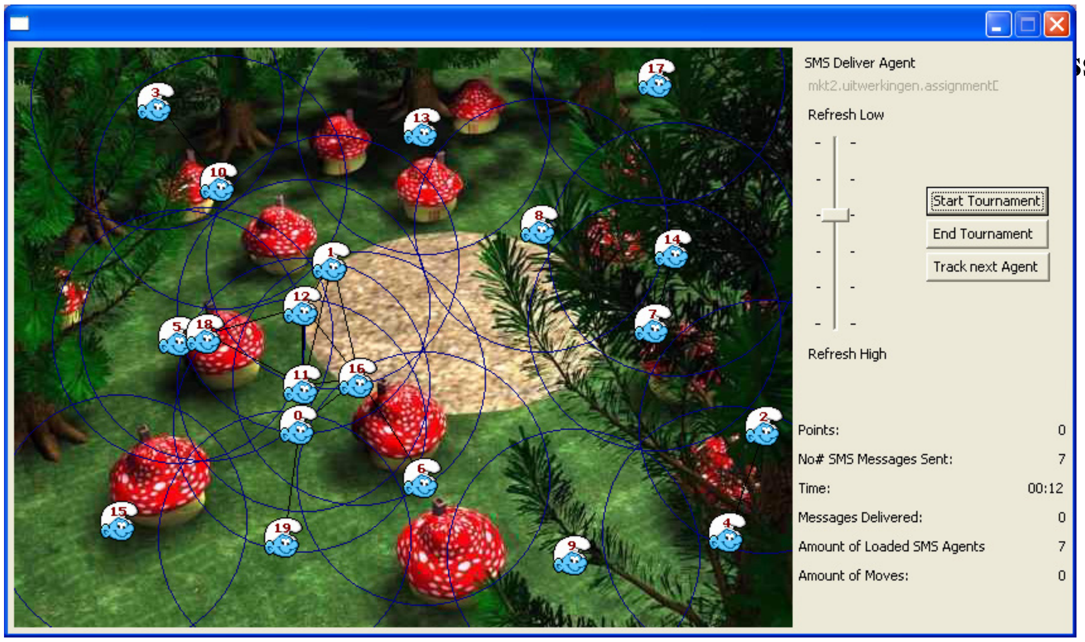
Design and implement an agent which uses the mobile agent concept, routes SMS messages to their destination in an ad-hoc network and optimizes the number of points earned. Points are only awarded for delivered messages. They are deducted, however, in any of the following situations: if a message failed to be delivered, for each transfer of a message from one node to another, and for each time unit used for the delivering of a message. Create a list of pitfall scenarios in which your agent should continue to properly function and use Fleeble to build an agent which follows the defined specifications. Explain the choices that you have made.

The assignment on ad-hoc networking has been designed around the concept of sending Short-Message-Service (SMS) messages via cell phones to other people in proximity. The environment described in the exercise is a park (see Fig. 2), where people (smurfs) walk around from one (random) location to the other. On regular intervals, an SMS message is sent from one random person to another. The person (smurf) does not pay his mobile network provider for the service. Instead, he sends the SMS over the ambient short-range network of mobile devices. The information available to students is as follows:

- Each node (smurf) is aware of its position in the park and of the direction it is heading.
- Each node identifies other nodes within its transmission range.
- Each node walks from one random location in the park to another. When the position is reached, another random location is chosen.

Two agents, each representing a possible non-optimal solution to this assignment, are given to the students in the beginning of the course. These agents are:

- *Reflex agent:* The reflex agent waits at the point of origin and periodically requests the identification of the nodes within the transmission range. If the target node comes into transmission range of the reflex agent, the agent delivers its message. This agent has a guaranteed benefit for each jump. Yet, messages will never be delivered if the target does not cross paths with the node of origin. In this setup, few points are lost for transfer of messages. However, due to inactivity of the agent, many points are lost on penalties for messages delivered too late.
- *Clone agent:* The clone agent takes a more aggressive approach to reach its goal. When the agent is present on a node which is not the target node, it requests the identification of the nodes within the transmission range. It then sends its own copy to all the neighboring nodes. These copies act in the same manner as the original. If the agent is present on the target node it delivers its message and it terminates its actions. Although the agent will always reach any target node connected to the network, it will use many points to transfer the copies of the messages from one node to another. In this setup many points are lost for transfer of messages. Even though this algorithm delivers messages quickly, the large amount of penalty points for transfer of messages makes this design a sub-optimal solution.



Points: 0
No# SMS Messages Sent: 7
Time: 00:12
Messages Delivered: 0
Amount of Loaded SMS Agents 7
Amount of Moves: 0

FIGURE 2: THE VISUAL INTERFACE OF THE ASSIGNMENT ON AD-HOC NETWORKING

The assignment does not have a perfect solution. The students only have to find a solution which results in a score being higher than a given minimum score. The students are told that the solution of the exercise is a trade-off between the reflex agent and the clone agent. They are also given a hint that a solution in which the agent copies itself only a limited number of times by using a hop-counter would represent a more optimal solution. Furthermore, the agent should use the positional information to locate their target node. Although an agent can 'see' its target node only if that node is in its transmission range, a system of agents can propagate the knowledge that they have gathered about positions of various nodes or they can store this information on nodes they traversed. This effectively increases the agent's scanning range. The information about the position of the nodes can be used to create a heuristic, which guides the agent to the node which is most likely to be in the vicinity of the target node. The transfer of information by means of recording the routing information on the traversed nodes is a variant of the ant-based routing [10]. Students should be aware that the information stored on various nodes is not always relevant; only recently stored information should be taken into account.

Students are also encouraged to optimize their solution for a competition which is held at the end of the course. In this competition, the teams of students compete against each. The group with the highest average score wins the competition and receives a symbolic prize.

Participating teams of students usually tackled this problem in the following way:

1. Describe the appropriate behavior of the routing agent given a list of pitfall scenarios in which the agent should continue to work properly. These scenarios may include:
   - The agent is loaded to a node and it received a message to be transferred to the target node. What should the agent do to achieve the goal in question?
   - The agent arrived to a node that is next to the target node. What must the agent do to deliver the message?
   - The agent arrived to a node that has no other nodes in transmission range except for the node from which the agent has arrived. What should the agent do?
   - The agent arrives at the target node. What should be done next?
2. Build an agent which inspects the surrounding nodes periodically. When the target node is in range, move to that location and deliver the message.
3. Adapt the agent so that it is able to copy itself a limited number of times and spread these copies to surrounding nodes. Use a counter (so-called hop-counter) to count the copies that are made. This ensures that only a limited number of copies are made.
4. Adapt the agent so that it is able to record the information it gathered on its path onto the nodes it currently traverses.
5. Adapt the agent to use the information stored on the nodes. If the information about the target node is available, find the node that is closest to the last location of the target node and move to that node.

CLASSROOM ASSESSMENT

Evaluation of the utilized educational methods and materials by students is very important in the educational programs of Delft University of Technology. Based upon the feedback provided by students, the utilized tools and readings can be improved to fit better the current generation of students, their preferences, and the curriculum goals of the educational program in question. A standard questionnaire is available for eliciting students' opinions on computer science courses, including all the relevant questions about the suitability of the used tools and readings, the parts of the course that taught them most, the overall educational experiences of the student, and the ways the course could be enhanced.

Already in 2002, 67% of the students who filled out the questionnaire indicated that the utilized agent framework is suitable for the goals of the course [6]. In 2003, even 89% of the students who filled out the questionnaire claimed so [6]. In 2004 and 2005, this percentage reached 92% [3].

The students judged the assignment on ad-hoc networking as being interesting and motivating. They claimed that they acquired new, valuable, and applicable knowledge concerning the concepts of mobility, distribution, and ad-hoc networking. They indicated that most of the learning occurred with the implementation of the solution to the posed problem.

In order to evaluate more objectively the educational benefit of the AI programming experience obtained during the coursework presented in this paper, we performed another assessment. We compared the performance of students' giving a presentation about ad-hoc networks and agent mobility and the performance of students' giving a presentation about the other subjects thought by the course (search algorithms, rule-based reasoning, semantic networks, predicate logic, epistemic logic, and knowledge acquisition). The presentations about ad-hoc networks and agent mobility rated the highest, 8,6 on a scale from 1 to 10, while the scores for the presentations about other topics had a range from 5,5 to 7,9. These results indicate that the students were interested to learn about agent mobility and ad-hoc networks, and that the programming assignment described above improved their ability to comprehend and verbalize the underlying concepts. In turn, these results also suggest that the presented coursework is highly suitable for teaching students the concepts in question – ad-hoc networks, mobility, and distribution.

CONCLUSIONS

This paper describes a method of teaching Agent Technology and Ad-hoc Networks using a novel, simple agent framework.

The developed simple agent framework, that we named *Fleeble*, supports the development of MAS applications and it embodies the concepts of concurrency, multi-agency, persistency, distribution and mobility. In contrast to the agent frameworks developed elsewhere, and as indicated by classroom experience, Fleeble is a suitable tool for teaching AI programming to novices because: (i) it is simple, it is accompanied by readable documentation, and it provides readily available, useful examples; (ii) it embodies simple, self-explained, visual interaction with the user that may concern one, more, or all currently running agents (Fig. 1). We expect that Fleeble can be widely useful for many different AI courses, including courses for non-computer-science majors.

The coursework presented in this paper represents a synthesis of the objectivist and the constructivist approach to introducing students to concepts of mobility, distribution, and ad-hoc networking. More specifically, the coursework builds on the theory of AI by teaching students how to use MAS and ad-hoc networking for solving a well-defined assignment (objectivist approach) aimed at sending SMS messages across an ad-hoc network (constructivist approach). Classroom experience indicates that the implemented pedagogy forms an effective way of teaching these "hot topics" in AI to novices and that the utilized "hybrid" teaching method increases the extent of learning compared to use of only the objectivist approach to teaching programming to novices.

ACKNOWLEDGMENT

The authors would like to thank all 210 MKT undergraduates who evaluated the course in the period 2002 – 2005. The work of Maja Pantic is supported by the Netherlands Organization for Scientific Research (NWO) Grant EW-639.021.202.

REFERENCES

[1] www.wi-fi.org (last visited: April 21, 2005)

[2] www.bluetooth.org (last visited: April 21, 2005)

[3] Pantic, M., Grootjans, R.J., Zwitserloot, R., "Fleeble Agent Framework for teaching an introductory course in AI", *Proc. Int'l Conf. Cognition and Exploratory Learning in Digital Age*, pp. 525-530, Lisbon, Portugal, 2004.

[4] McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y.B-D., Laxer, C., Thomas, L., Utting, I., "Multi-national, multi-institutional study of assessment of programming skills of first-year CS students", *SIGCSE Bulletin*, vol. 33, no. 4, Apr. 2001, pp. 1-16.

[5] Pantic, M., Zwitserloot, R., Grootjans, R.J., "Simple Agent Framework: An educational tool introducing the basics of AI programming", *Proc. IEEE Int'l Conf. Information Technology in Research and Education*, pp. 426-430, Newark, USA, 2003.

[6] Pantic, M., Zwitserloot, R., Grootjans, R.J., "Teaching Introductory Artificial Intelligence Using a Simple Agent Framework", *IEEE Transactions on Education*, Vol. 48, No. 2, May 2005, pp. xx-xx.

[7] Ho, V., Wobcke, W., Compton, P., "EMMA: An E-Mail Management Assistant", *Proc. IEEE/WIC Int'l Conf. Intelligent Agent Technology*, pp. 67-74, Halifax, Canada, 2003.

[8] Gao, X., Zhang, M., "Learning knowledge bases for information extraction from multiple text based Web sites", *Proc. IEEE/WIC Int'l Conf. Intelligent Agent Technology*, pp. 119-125, Halifax, Canada, 2003.

[9] Ha, S.H., "Helping online customers decide through Web personalization", Intelligent Systems, Vol. 17, No. 6, Nov/Dec 2002, pp 34-43.

[10] Di Caro, G, Dorigo, M., "AntNet: Distributed Stigmergetic Control for Communications Networks", Journal of Artificial Intelligence Research, vol. 9, 1998, pp. 317-365.