Agent-mediated Online Learning - AMOL

Xun Yi, Chee Kheong Siew

Information Communication Institute of Singapore
School of Electrical & Electronic Engineering,
Nanyang Technological University
Blk S2, Nanyang Avenue, Singapore 639798

Mahbubur Rahman Syed

Department of Computer and Information Sciences
Wissink Hall 273, Minnesota State University, Mankato
MN 56001, USA

Abstract

The growth of intelligent agent technology allows new developments in the way learners obtain knowledge on the Internet. This paper proposes an agent-mediated online learning architecture, called AMOL, to automate online learning processes.

Keywords: ALN, Adele, Intelligent agents, Online learning

1. Introduction

Asynchronous Learning Networks (ALN) are networks of people who can learn anytime anywhere. ALN combines self-study with substantial, rapid, asynchronous interactivity with others. In ALN learners use computer and communications technologies to work with remote learning resources, including coaches and other learners, but without the requirement to be online at the same time. The most common ALN communication tool is the World Wide Web.

By this definition, a web-based workshop that requires frequent online conferencing and collaboration with others is ALN. So is a text- or computer-based training course that requires learners to use email to discuss assignments with each other and with the coach. ALN also encompasses a proctored examination at a specified time and place, or occasional synchronous chat or lab sessions for near-campus learners, or an in-person kickoff meeting.

By this definition, distance education based primarily on a synchronous audio or video presentation or conference is not ALN because these constantly require learners and instructors to be available at the same time. A videotaped course or mail-based correspondence course or computer-based training is not ALN because these do not include substantial and rapid interactivity with others, even though the learner might mail in a paper or test and receive a reply days later.

There are many challenges facing ALN. In distributed learning environments where there is the potential for losing the cohesiveness and spontaneity of the classroom experience, it is essential to understand how to improve the online learning experience so that it approaches and perhaps even exceeds more traditional learning methods. The instant availability of a human tutor online would be ideal. However, providing this capability is no more realistic than continuously providing a human tutor for the traditional classroom-based learning experience. Cost and availability are limiting factors in supplying continuously attentive human tutors. Often students simply want questions answered and would be happy with any type of effective immediate feedback – human or machine. An augmented anytime capability is particularly important in learning environments in which online tutors may not be available for extended periods (e.g., due to differences in time zones or to late-night student study habits).

ALN courses can be improved by the introduction of autonomous intelligent agents [1]. Intelligent Agents (IAs), termed "knowbots" (or Knowledge Robots) can perform the duties of online facilitators for routine tasks. Checking computer code, responding to simple questions, reminding learners about the need to turn in assignments and potentially even grading essays are among the types of things that intelligent agents can accomplish. The feedback can be provided by intelligent agents in an on-demand format for certain types of information requirements.

In contrast to the above Web-based intelligent agent system where "knowbots" sits on the server side, resulting in increased latency in tutor response to student actions, the Adele (Agent for Distance Education – Light Edition) [2], a pedagogical agent, implements key pedagogical functions: presentation, student monitoring and feedback, probing question, hints, and explanations. These capabilities are coupled with an animated persona that supports continuous multi-modal interaction with a student. The architecture supports client-side execution in a Web browser environment, and is able to inter-operate with simulations created by off-the-shelf authoring tools.

Most of the existing intelligent agent online learning systems only focus on applying static intelligent agents that reside at Web servers or Web browsers in solving distance education problems. For instances, "knowbots" and Adele. System designers have ignored another key attribute of intelligent agents, that is, their mobility.

In this paper, we focus on the use of the mobility of intelligent agents in automating online learning process and proposed an agent-mediated online learning (AMOL) architecture. Under AMOL scenario, (1) online learning is able to choose most suitable study material or ongoing or coming conferencing systems with the aid of searching intelligent agent; (2) pedagogical agents residing at online learners' PCs can automatically gather online teaching material with the aid of gathering intelligent agents; (3) once pedagogical agents cannot answer students' questions, it will automatically try to obtain solutions with the aid of querying intelligent agents.

The following sections are arranged as follows: Section 2 introduces intelligent agent technology; Section 3 describes the agent-mediated online learning (AMOL) architecture; Conclusion is drawn in the last section.

2. Intelligent Agent

Intelligent agents are one of the fastest growing areas of information technology. They offer a new paradigm for developing software applications. More than this, agent-based computing has been hailed as "the next significant break-through in software development" [3], and "the new revolution in software" [4]. Agents are being used in an increasingly wide variety of applications, ranging from comparatively small systems such as Email filters to large, open, complex, mission critical systems such as Internet trading.

General Concept of Intelligent Agent

An agent's role is of one action on behalf of others. In the field of artificial intelligence this term is used to refer to an entity that functions autonomously in a particular environment and provides service to its owner. This agent is autonomous in the sense that its activities do not need human intervention. People have long dreamed of intelligent system which can let them "input less and output more" to replace traditional ordinary applications with little flexibility that depend on step by step learnings to be input. Intelligent agent research is trying to narrow the gap between the dream and the reality. The hope is that agents will provide services to people without requiring them to explicitly indicate the procedures required, make appropriate decisions in unexpected or novel environments, plan their action in advance and tackle problems independently.

The term "agent" has its background in the early artificial intelligent approach to the humanoid entities. During the long period of development of artificial intelligence the term was applied to a wide range of fields and attracted different definitions. Genesereth [5] regards agents as "software components that communicate with their peers by exchanging messages in an expressive agent communication language". Shoham [6] regards agents as being ontologically dependent on their role in the community.

An agent can be thought of as a computer program that simulates a human relationship by doing something that another person could do for you [7]. An agent is a self-contained program capable of controlling its own decision making and acting, based on its perception of its environment, in pursuit of one or more objectives. More than one type of agent is possible. In its simplest form it is a software object that sifts through large amounts of data and presents a subset of this data as useful information to another agent, user or system. An example of this is an agent that reads and analyzes all incoming e-mail, and routes it to an appropriate department or another agent for a reply [8]. These types of agents are called static agents.

Mobile agents owned by a user or another software element are capable of migrating from one computer to another to execute a set of tasks on behalf of their owner. Such agents would typically gather and analyze data from a multitude of nodes on the network, and present a subset of this data as information to a user, agent or system. For example, a company which needs to order additional paper supplies could have agents monitoring the quantity and usage patterns of paper within the company and launching buying agents when supplies are becoming low. Those buying agents automatically collect information about vendors and products that may fit the needs of the company, evaluate the different offerings, make a decision about which merchants and products to pursue, negotiate the terms of transactions with these merchants and finally place orders and make automated payments. Mobile agents can also act as brokers for users. For example, a single sign-on agent can sign-on to many different systems relieving the user from typing in his/her password for every system [8].

Mobile agents are said to be autonomous, in the sense that they can make their own decisions while away from their host. This implies that a mobile agent (agent, for short) is not just a piece of data being transferred between systems, but may also carry some logic (i.e. code) and state, which enables it to perform some part of its tasks in one system, migrate to another and continue its work there. In this paper we will mainly focus on a discussion of mobile agents.

Mobile Agent System

A mobile agent framework naturally draws upon existing components of networks such as name servers, directories, routers and so forth, and adds several new facilities:

- Mobile Agents

  These may be expressed in various procedural languages and may transport knowledge expressed in various forms. They must be structured so as to engage in a progressive dialogue with the Agent Meeting Places (AMPs) until they are able to execute or are rejected.

- Agent Languages

  Two kinds of languages are involved. One is the language in which the programmatic content of the agent is written; this is usually (though not necessarily) a script language. The second is a language for knowledge representation, which provides the means to express goals, tasks, preferences, and vocabularies appropriate to various domains.

- Agent Meeting Places

  These have various subcomponents, and they are the principal means by which a server becomes part of the agent framework. We think of the AMP as a broker between the agents that are making requests for resources and services and the applications that implement these resources and services.

- Public Security Services

  These security services, which are trusted by the servers taking part in the agent framework, provide certificates of authenticity and other security services to the mobile agents.

Operation of Mobile Agent System

In many cases, the mobile agent is launched from a client device such as a laptop or desktop PC by an otherwise conventional application. We anticipate that end-users will not in general write their own agents (though that is certainly possible), but that various classes of agents will be distributed by services for use by their subscribers or will be packaged with client applications. The agent is initialized with the user's task and transmitted by a message channel. The sending client may specify a destination service directly, but more than likely it sends the agent initially to, for example, a Yellow Pages server, which can propose servers to be visited that are likely to be able to fulfill the user's task.

When the agent reaches a server, it is delivered to an Agent Meeting Point (AMP). Upon arrival at an AMP, the Agent Passport is inspected for its authentication credentials. Once validated, the AMP examines the agent's Table of Contents (TOC). Ontologically named service requests are resolved to determine if the desired services are available at the AMP. If sufficient resources are available at the AMP and are permitted to the agent, the constituent parts of the agent are passed to the services.

The executable portions of the agent are then started. In some cases, the mobile agent will be interacting directly with server resources (via proxy objects, which enable access control to be enforced). In the other cases, the mobile agent will interact with a static agent at the AMP. A static agent is an agent that is resident at that AMP. The static agent may have been installed by the server operator. Static agents enable the server's function to be personalized by the server's owner or by users. When the agent has successfully completed its task at this server, it may ask to be transported to a new host. Alternatively, it may launch a smaller agent to deliver the acquired information to the sending client or to another server and then it may terminate. This ability to acquire knowledge and transport it from place to place is a key attribute of a mobile agent. The new knowledge may simply a new destination or it may be a security token or a transaction. This ability means that the agent is not merely a program that is executed on a remote host and then returned to its origin, but it is a dynamic process which progressively accomplishes a task by moving from place to place.

If the agent proves to be unauthorized, or if the meeting point is unable to provide the agent with the resources it has requested, the AMP will take action based on the agent's header. It may discard the agent; send the indicated party a description of the failure; or if it is capable, propose one or more AMPs that may be able to satisfy the request.

3. Agent-mediated Online Learning (AMOL) Architecture

Our agent-mediated online learning (AMOL) architecture aims at automating online learning process by use of mobile agents. AMOL is apply to the following scenario:

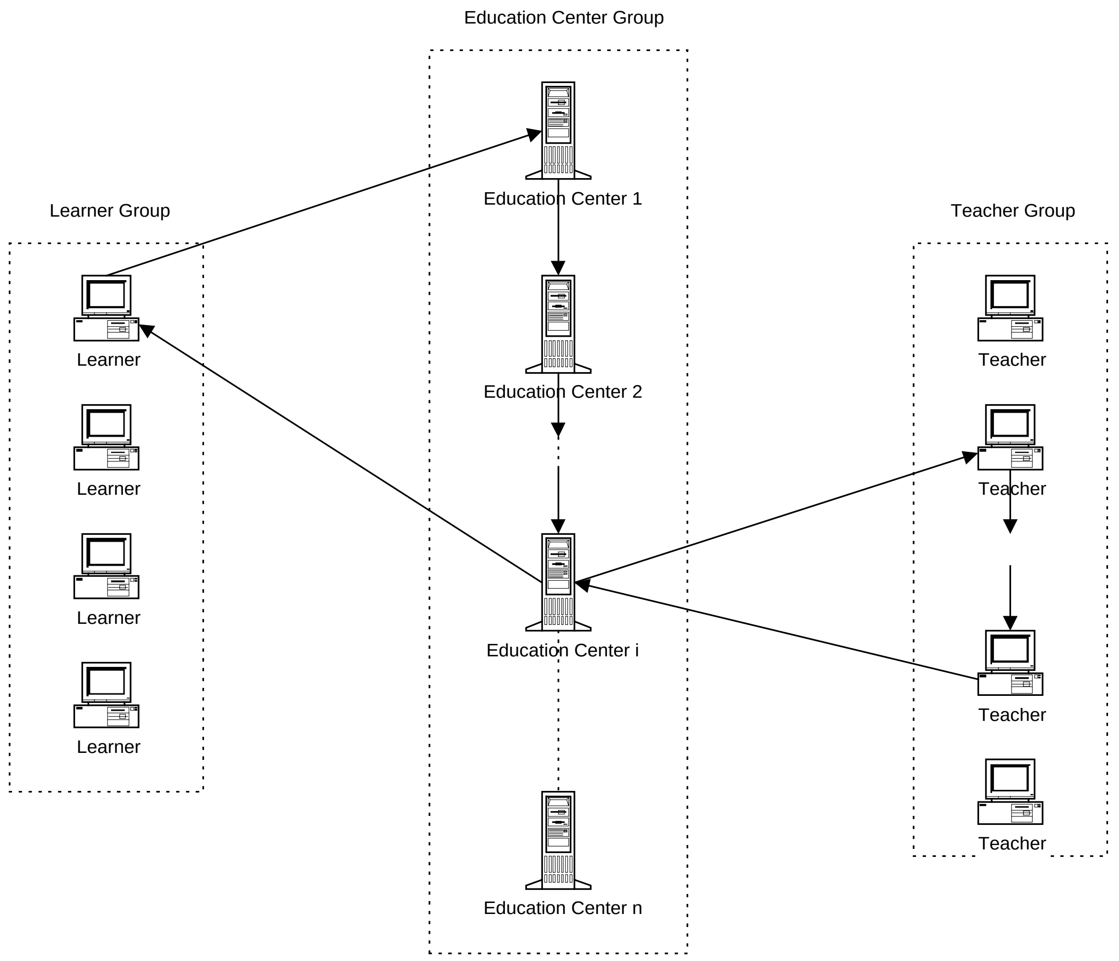

Figure 1: Agent-mediated online learning (AMOL) architecture

As shown in Figure 1, AMOL scenario consists on three parties of participants:

1. Learner Group: the set of learner who cannot attend conventional classes due to distance or time constraints, but have personal computers connecting to the Internet and try to learn some online material.

2. Education Center Group: the set of education center servers which provide online study material, assignment repository, “knowbots”, knowledge databases, various pedagogical agents, and etc.

3. Teacher Group: the set of teachers who can answer students’ questions and check the assignments for students, and possess personal computers connecting to the Internet.

AMOL can be described in the following phases.

Software Installation

In order for pedagogical agents to run, learners have to install necessary software on their computers to support mobile agent language. In AMOL, the Aglets Software Development Kit [9] should be installed on the computer of each learner.

The Aglet Software Development Kit is an environment for programming mobile Internet agents in Java^{TM} (it used to be called the Aglets Workbench). The Java Aglet Application Programming Interface (J-AAPI) is a standard for interfacing aglets and their environment. J-AAPI is simple, flexible, and stable. Internet agent developers or agents can write platform independent aglets and expect them to run on any host that supports J-AAPI.

The above process can be regarded as establishing “tutorial rooms” for online learners.

Searching Pedagogical Agent

Assume that an online learner want to study some online material. He creates a searching intelligent agent and tells it his requirements for the pedagogical agents or conferencing systems. These requirements encompasses:

1. Language of study material

2. Topic of study material

3. Level of study material (for example, primary, secondary or high levels)

4. Type of study material (for example, text, audio or video)

5. Time of study (in case of searching conferencing systems)

6. Others

Then, the searching intelligent agent is launched from the PC of the online learner. It automatically roams among a series of online education centers and searches the most suitable online material.

In case of searching conferencing systems, once suitable ongoing or coming conferencing systems matching the requirements of the online learner are found, the searching intelligent agent reports its findings to the online learner and then automatically lapses.

In case of searching online material with pedagogical agent, once suitable online material with pedagogical agent matching the requirements of the online learner is found, the searching intelligent agent invites the pedagogical agent to the online learner's PC and then automatically lapses.

As shown in Figure 1, the searching intelligent agent finds what it wants at Education Center i and then lapses there.

Pedagogical Agent Residing

As the searching intelligent agent indicates, a pedagogical agent at Education Center i brings necessary teaching material to the online learner and resides on the PC of the online learner. In AMOL, the pedagogical agents are supposed to be animated pedagogical agents.

AMOL architecture on the PC of an online learner can be depicted as:

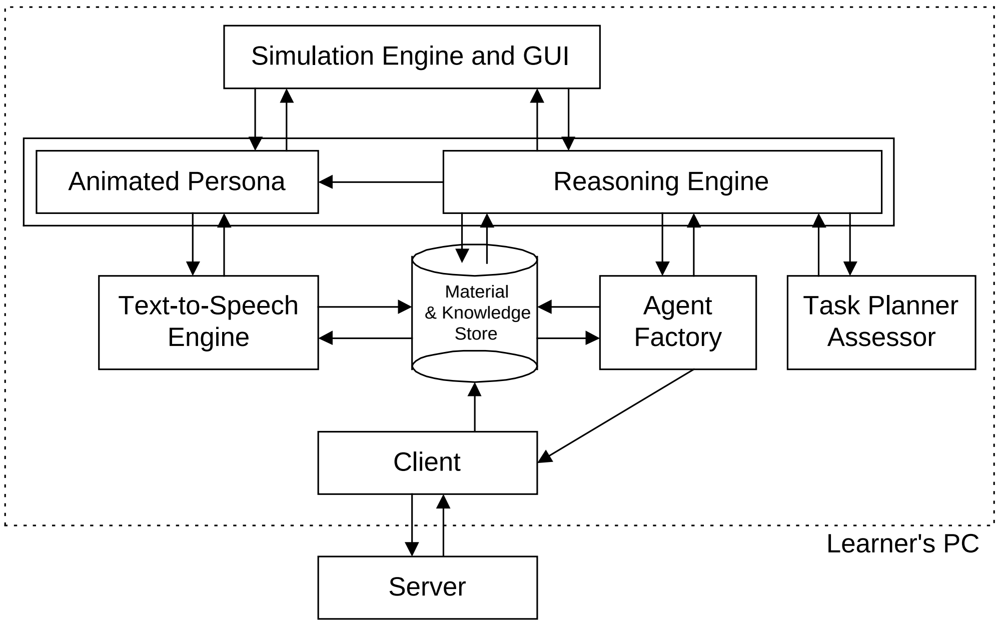

Figure 2. The structure of AMOL architecture in a learner's side

As shown in Figure 2, AMOL architecture in a learner's side consists of four main components: the pedagogical agent, the simulation, the client-server, and the Material & Knowledge Store. The pedagogical agent consists further of two sub-components, the animated persona and the reasoning engine. Text-to-speech engine and agent factory are installed on the PC of an online learner during software installation.

The reasoning engine performs all monitoring and decision making. Its decisions are based on a learner model, a case task plan, and an initial state, which are determined at the Education Center i when the searching intelligent agent interacts with the Education Center i, and on the agent's current mental state, which is updated as a learner works through a case. The record of the learner's actions is saved in the Material & Knowledge Store where it is used to assess the level of the learner's expertise and determine how the pedagogical agent will interact with the learner in future cases.

The animated persona is simply an Aglet that can be used alone with a Web page-based JavaScript interface or incorporated into larger applications, such as the simulation-based exercises. The persona applet allows animation frames to be easily added and exchanged to support a choice of personas.

The simulation can be authored using the language or authoring tool of one's choice. All simulations communicate with the agent via a common application programming interface (API) that supports event and state change notifications.

The integrated system is downloaded to and run on the client's side for execution efficiency. This is in contrast to the architecture of most other Web-based Intelligent Tutoring Systems where the intelligent tutor sits on the server side, resulting in increased latency in tutor response to learner actions [10]. Reducing latency is especially critical when one considers animating an agent's response to a learner's action, in order to achieve the perception of awareness in a shared workspace.

Tutoring

After the pedagogical agent resides on the PC of the online learner, it performs tutoring on basis of task plan and feedback.

Depending on the context, a task can be a simple sequence of steps or it can be a complex and non-linear partial ordering on a set of steps. As Adele, AMOL represents all procedural tasks using a standard hierarchical plan [11]. A plan hierarchy is comprised of steps, each of which is either a primitive action (e.g. corresponds to a simulation event) or a complex action (e.g. is itself a plan).

Preconditions and end conditions are represented by Boolean expressions in conjunctive normal form. Steps are supported as goals by sub-classing the expression tree to include a new type of Boolean, a step expression, which, like Boolean constants, logical expressions, and comparative expressions, evaluates to true or false. Evaluating a step expression is equivalent to evaluating a step's end conditions, which is done recursively in the case of complex plans. This extension makes it possible to support the creation of both goal- and task-based plans.

The plan hierarchy is evaluated at each step to account for the dynamic nature of a simulation and the unpredictability of a learner's actions. Actions whose goals become 'undone' are automatically re-executed while those whose goals are implicitly satisfied are skipped. In this way, AMOL's task plan is dynamically updated.

The reasoning engine runs in three modes. In its most restrictive mode, it will simply block actions whose preconditions are unsatisfied. Same as Adele, AMOL uses this opportunity to provide unsolicited feedback about what should be done to satisfy the desired step's preconditions. The persona displays a Hint button so that a learner may also ask for hints directly, before guessing or taking an incorrect action. (Similarly, the persona has a Why? button, that allows a learner to ask for a rationale.) In practice mode, the engine does not block - the learner can make mistakes - and does not provide unsolicited feedback, but still allows a learner to ask for hints.

Once the teaching material available at the Material & Knowledge Store is not enough, the pedagogical agent will generate and launch a gathering intelligent agent which roams among the education centers, gathers related teaching material and brings them back.

Querying

Learning process is interactive. With the limited knowledge store at the learner's side, the pedagogical agent may not be able to answer various questions and judge various answers from learners. In this case, we can consider making use of those "knowbots" residing at distributed education centers. For this purpose, the pedagogical agent creates and launches a querying intelligent agent from the online learner's PC. The querying automatically roams in a series of online education centers and interacts with those "knowbots" to solve problems as shown in Figure 3.

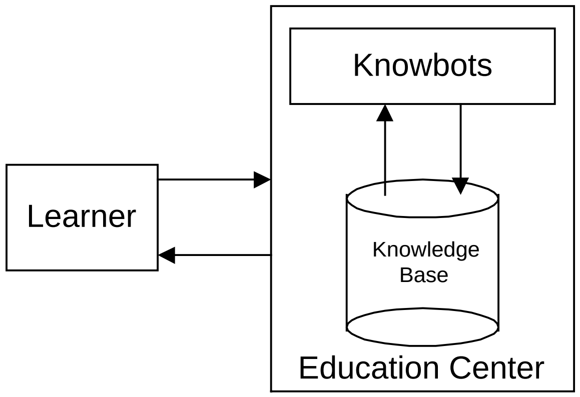

Figure 3. Querying system of AMOL architecture

If the querying intelligent agent cannot solve problems with the aid of knowledge bases of online education centers, it visits a series of online teachers' PCs as shown in Figure 1. Why do we need the querying intelligent agent to visit online teachers' PCs one by one instead of broadcasting queries to them? The reason is to avoid repetition of teachers' works.

Once the querying intelligent agent gets solutions, it immediately sends them to the pedagogical agent in the learner's side and then automatically lapses.

Terminating

Once the pedagogical agent accomplished its teaching mission at the PC of the online learner, it erases all data in the Material and Knowledge Store and brings the study record of the online learner back to the Education Center i. Then Education Center i save the study record and study period into a database for future use.

4. Conclusion

Most of the existing agent-based online learning systems only focus on using static agent to solve only learning problems and ignore the mobility attribute of intelligent agent. Distinguishing from the traditional online learning systems with single education center, an agent-mediated online learning (AMOL) architecture under which multi-systems for online learning collaborate to automate online learning processes has been proposed above. AMOL made full use of the mobility of intelligent agent. Mobile agents in AMOL can be classified into: pedagogical agent, searching agent and querying agent.

The advantages of AMOL lie at: (1) online learning is able to choose most suitable study material or ongoing or coming conferencing systems with the aid of searching intelligent agent; (2) pedagogical agents residing at online learners' PCs can automatically gather online teaching material with the aid of gathering intelligent agents; (3) once pedagogical agents cannot answer students' questions, it will automatically try to obtain solutions with the aid of querying intelligent agents.

In our future work, we will proceed to implement AMOL architecture by using Aglets Software Development Kit. Interested parties are welcome to comment on AMOL and offer valuable advices about implementation.

References

[1] Thaiupathump, C., Bourne, J., Campbell, J. O. (1999) Intelligent Agents for Online Learning. Journal of Asynchronous Learning Networks, Volume 3, Issue 2, November 1999.

[2] Shaw, E., Johnson, W.L., and Ganeshan, R. (1999) Pedagogical Agents on the Web. Proceedings of the Third Int'l Conf. on Autonomous Agents, pp. 283-290, May, 1999.

[3] Sargent, P. (1992) Back to school for a brand new ABC. In The Guardian, 12 March. P.28

[4] Ovum Report (1994) Intelligent agents: the new revolution in software.

[5] Genesereth, M. R., Ketchpel, S. P. (1994) Software agent, Communications of ACM, 37(7), 48-53.

[6] Shoham, Y. (1993) Agent-oriented programming. Artificial Intelligency, 60(1), 51-92.

[7] Selker, T. (1994) A teaching agent that learns, Communications of the ACM, 37(7).

[8] Wirthman, L. (1996) Gradient DCE has sign-on feature, PC Week, March 1996.

[9] http://www.trl.ibm.co.jp/aglets/

[10] Brusilovsky, P., Schwartz, E., and Weber, G., ELM-ART: An intelligent tutoring system on world wide web. Frasson, C., Gauthier, G. and Lesgold, A. (Eds.), Proc. of the Third Int'l Conf. on Intelligent Tutoring Systems, pp. 261-269, Springer Verlag, 1996.

[11] Russell, S., and Norvig, P., Artificial Intelligence: A Modern Approach. Prentice Hall, 1995.