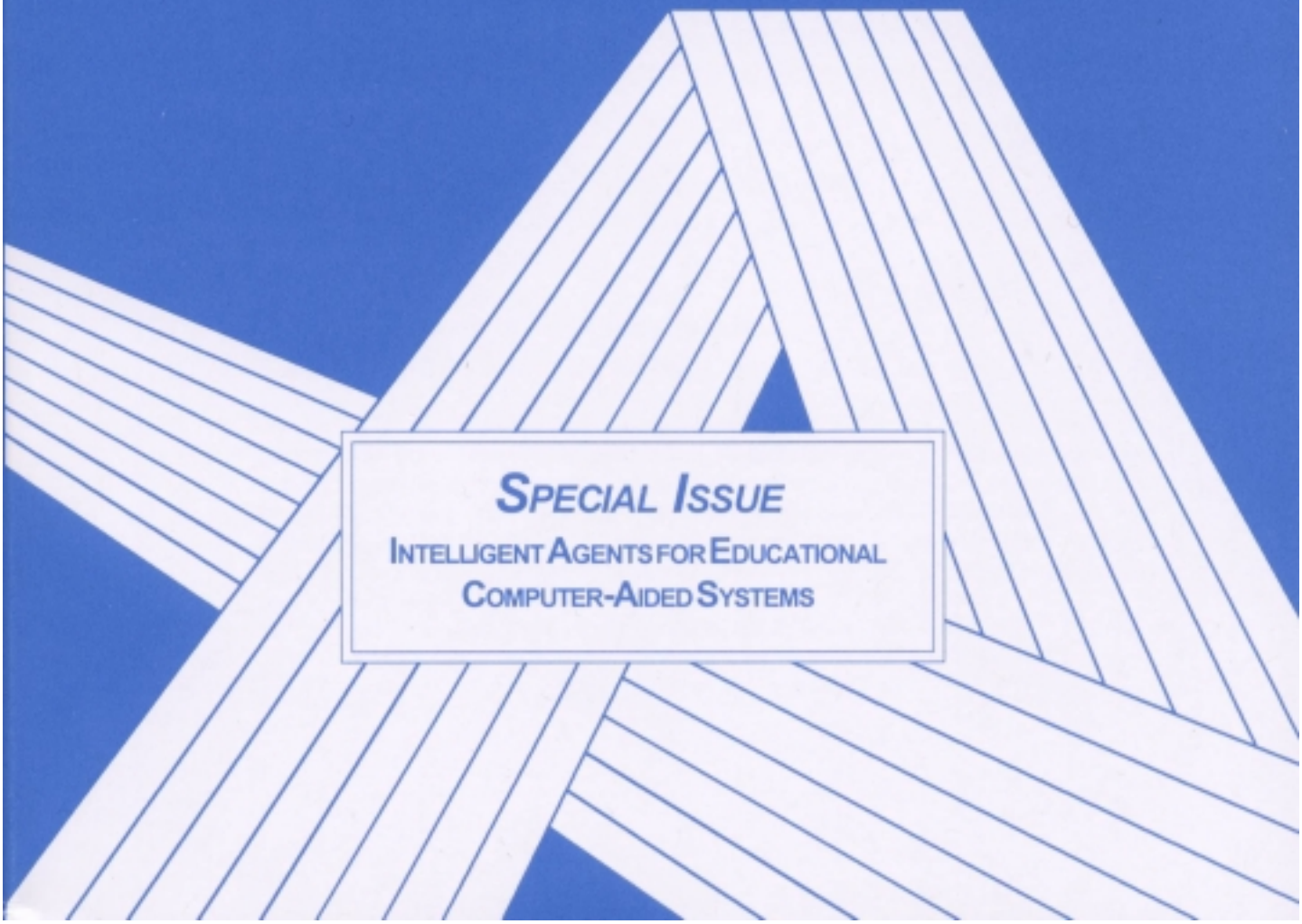
VOL. 10 NO. 3/4 1999

Journal Of Interactive Learning Research

FORMERLY JOURNAL OF ARTIFICIAL INTELLIGENCE IN EDUCATION



SPECIAL ISSUE

INTELLIGENT AGENTS FOR EDUCATIONAL COMPUTER-AIDED SYSTEMS

JOURNAL of INTERACTIVE LEARNING RESEARCH

Volume 10, Number 3/4 1999

Special Issue on Intelligent Agents for Educational Computer-Aided Systems

Articles

The *Journal of Interactive Learning Research* (ISSN 1093-023X) is published quarterly by the Association for the Advancement of Computing in Education (AACE), an international, educational, nonprofit organization. Annual US membership/subscription rates: $75 individuals; $105 schools, libraries, and other institutions. Annual non-US membership/subscription rates, add: $10 for postage. US funds on US bank, Money Order, Master Card, or VISA. Single copy price: $20 plus the following shipping fees per book: $2.50—US; $3.50—Canada/Mexico; $4.50—elsewhere.

Publisher: AACE, PO Box 2966, Charlottesville, VA 22902, USA.
804-973-3987; Fax: 804-978-7449; Email: info@aace.org.
website:http://www.aace.org

Jl. of Interactive Learning Research (1999) **10**(3/4), 275-285

Agent-Based Computer Tutorial System: An Experiment for Teaching Computer Languages (ATCL)

MAHMOUD M. EL-KHOULY, BEHROUZ H. FAR,
AND ZENYA KOONO
Saitama University, Faculty of Engineering, Information and Computer Sciences Department, FAR Lab
T 338, Urawa-Shi, Shimo-Okubo, 255
Saitama-ken, Japan
{elkhouly,far,koono}@cit.ics.saitama-u.ac.jp

This paper presents a new vision for intelligent computer aided instuction (ICAI) in the presence of agent technology. An agent-based computer tutorial system consists of two sub-agents; (a) personal assistant agent for teachers (PAA-T), and (b) personal assistant agent for students (PAA-S). PAA-T that allows the teachers to cope with the knowledge base of a computer language under investigation, and add or modify the commands' structure that will be taught. Also, this agent can generate a new tutoring dialog for a new computer programming language by consulting previous tutoring dialogs for another computer programming language. PAA-S contains a student model and a tutoring module. In the student model, the system can accept free-format answers from the student, and check it against the language structure. Tutoring text has been separated from the tutorial module, such that the student's mother tongue can be used. The system is suitable for any computer procedural language (e.g., FORTRAN, PASCAL, etc.). The system has been tested in some schools, and the feedback has been taken into consideration. Using these kinds of agents allows us to expand their features to include communication with other agents and to exchange teacher's experiences as well as tutoring dialogs.

Learning one of the computer languages today, is essential for students in elementary private schools in Egypt, and in prep governmental schools. However, the lack of good instructors is a problem, the government provides each school with a complete laboratory containing one server and 20 PCs (clients) and the mathematician teachers must enroll in several courses of computer sciences to be able to teach computer languages in their schools. Using computer-aided instruction (CAI) can solve this problem, since it needs only few teachers to guide the students in the laboratory. Moreover, the material of the computer course varies from one year to the other, for example, the student who learnt BASIC last year, will learn FORTRAN this year. So, we need many CAI for different computer languages.

An ICAI system typically contains an expertise module, student model, and tutoring module. The most basic approach for a student model is called the "overlay approach" (Carr & Goldstein, 1977), in which the student model is assumed to be a subset of the expert model. The "coached problem solving approach" is a famous approach in the tutoring module, the initiative in the student-tutor interaction changes according to the progress being made. As long as the student proceeds along a correct solution, the tutor merely indicates agreement with each step. When the student stumbles on a certain part of the problem, the tutor helps the student overcome the impasse by providing tailored hints that lead the student back to the correct solution path. In this system, both "overlay approach" and "coached problem solving" are used in the student model and the tutoring module respectively.

Some agent based architectures for ICAI have already been developed (Kenneth & Sven, 1998; John & Claude, 1998; Ueno, 1994). However, their system suppose that the student has previous computer background knowledge. Moreover, the current commercial CAI systems in the market have some shortages of their modules, for example, in the student model they prefer to use multiple choice answers to get the reply from the students rather than a free format answer, since it is easy to be handled by the system. However, it can't give an accurate result about the students' knowledge. Another shortage is language dependence used in designing the system, which makes the system unsuitable for other countries that use different languages. In this research, we covered both shortages.

The proposed system consists of two sub-agents representing a server-client relationship, personal assistant agent for teachers (PAA-T) as a "server," and personal assistant agent for students (PAA-S) as a "client." First, a human expert produces the semantic rules for a computer programming language to be taught, then the PAA-T searches the previous tutoring text for some/all of these semantic rules. If it finds some/all then it produces the tutoring text base, otherwise it asks the expert to provide it. The PAA-S can

communicate with PAA-T to retrieve the tutoring dialog of the command(s) which a student wants to practice. PAA-S consists of tutoring module, student model, and student's behavior-base (as shown in Figure 1).

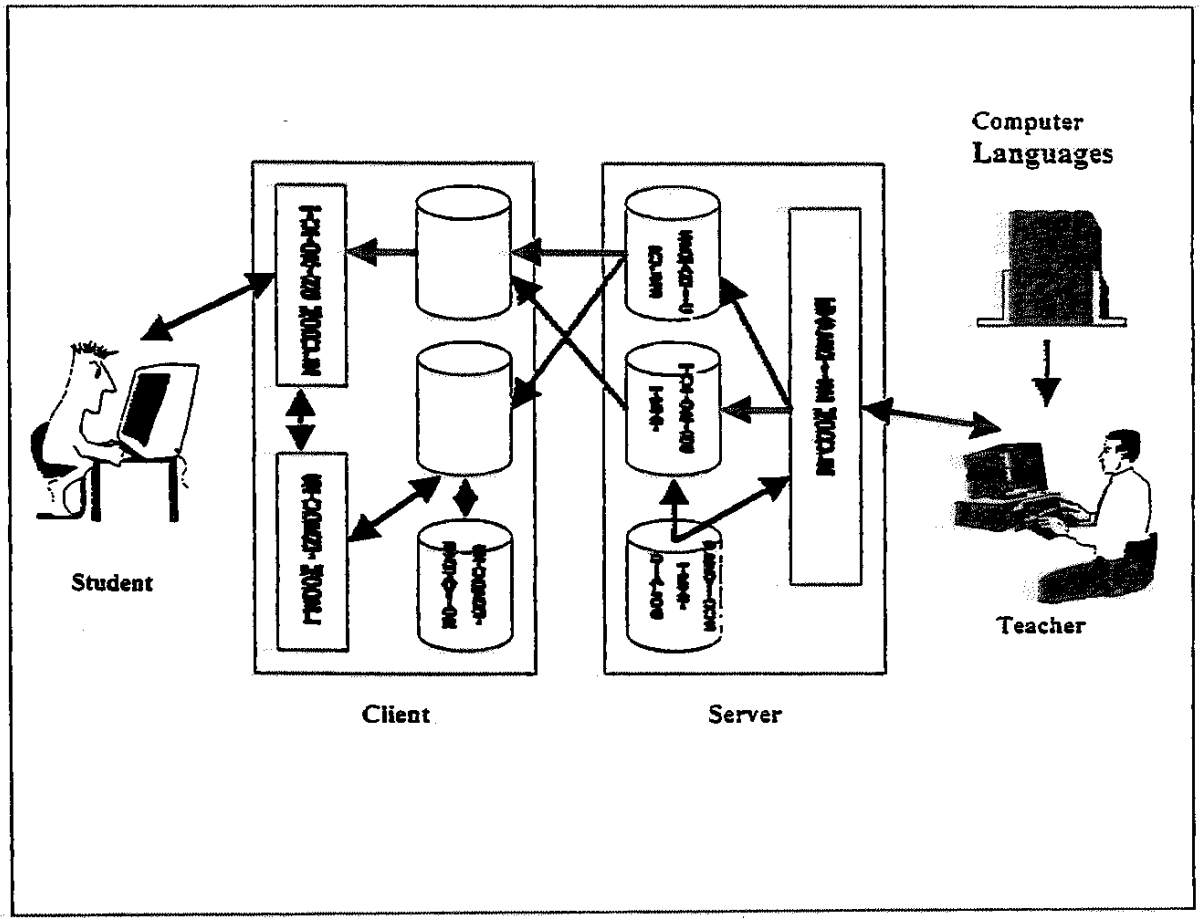


Figure 1. ATCL system

PERSONAL ASSISTANT AGENT FOR TEACHERS

The personal assistant agent for teachers (PAA-T) consists of four parts; expertise module, semantic rules base, tutoring text base, and previous tutoring text base. PAA-T helps the teacher cope with the knowledge base of a computer language under investigation, to add or modify the command's structure that will be taught, and to produce a meta level language representing this computer language, which we call it "semantic rules." A main reason for separating the PAA-T from the PAA-S, is that the system can accept different computer programming languages for PAA-T without affecting the mechanism of PAA-S.

Expertise Module

The expertise module consists of domain knowledge that the teacher intends to teach to the student. Representative Artificial Intelligence (AI) methods used to organize the domain knowledge in the expertise module include development of semantic networks, application of production systems, procedural representations, and building scripts-frames (Hartley & Tait, 1986). In this system, a production rule to construct modular representations of commands is used.

Semantic Rules

When the teacher wants to build the semantic rule base, the screen of building the semantic rules base will appear containing the keys guide to build the knowledge-base (see Figure 2).

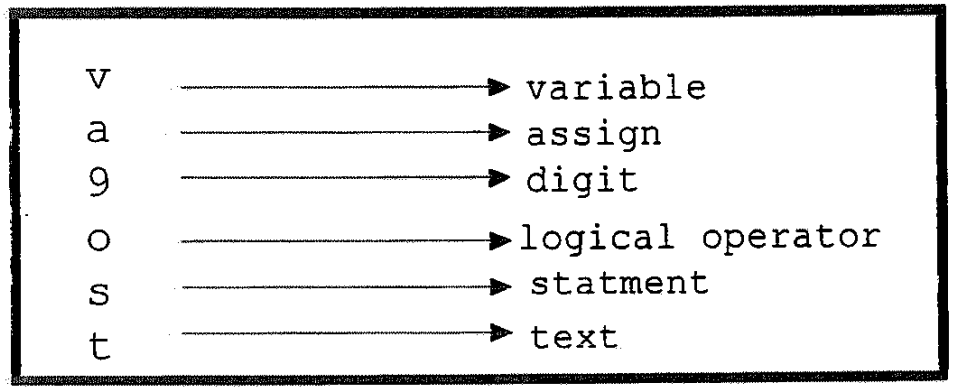


Figure 2. Keys guide to build semantic rules base

In this phase, the teacher puts the structure of each command (using upper case characters for reserved keywords only), for example, LET vav, means that one of the reserved keywords of BASIC language is called LET, and its structure is (Variable assign Variable). That is in a case when the command can be represented only by one reserved keyword, but if the command needs more than one reserved keyword to be represented, the teacher may enter it as in the following example:

IF vov THEN s,

which means that (if the condition (variable operator variable) is satisfied then do the statement s).

The tutoring module retrieves the structure of commands to be presented during the tutoring dialog. The student model consults it to check the answers of the students.

Tutoring Dialogs

Any computer language has command(s) for an In/Out (I/O), condition, loop,and so forth. The way to teach those commands is almost the same for all computer languages, for instance, to teach the condition command in BASIC will be done in quite the same way as the condition command in FORTRAN. The personal assistant agent for the teacher can construct the tutoring module for the new language by retrieving the tutoring dialog of similar commands in another language and adapts it. For example, if the agent has the tutoring dialog of how it can present the IF command in BASIC, then it consults the semantic rules of FORTRAN to substitute the BASIC command in the tutoring dialog with the FORTRAN command (see Figure 3).

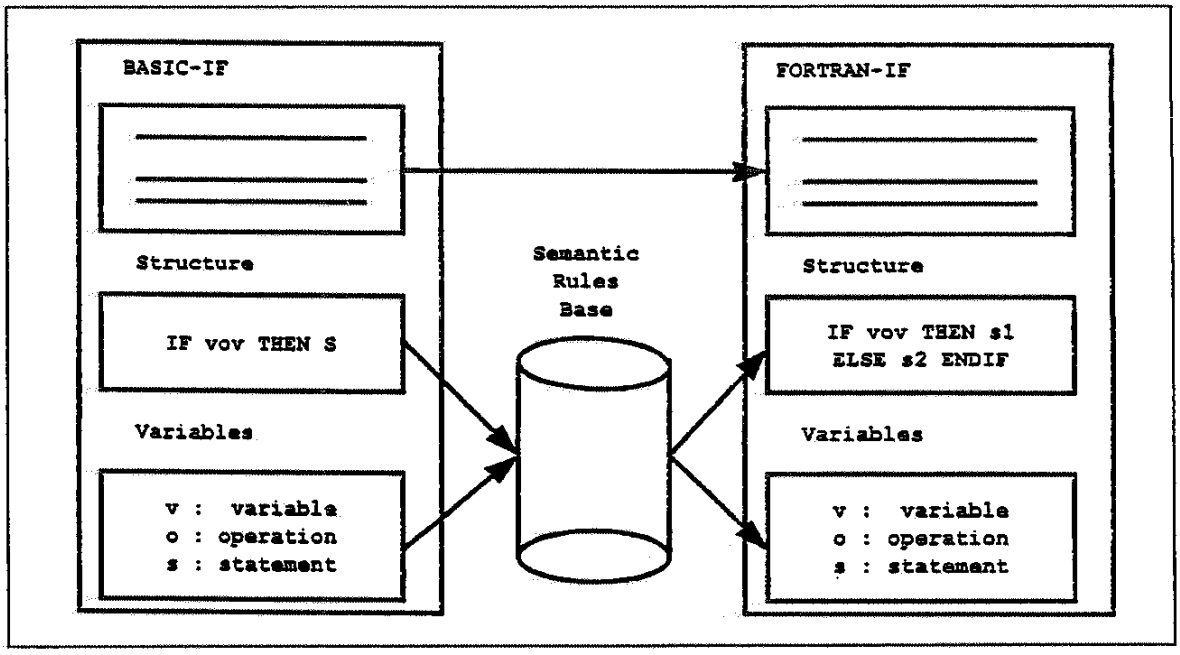


Figure 3. Adapting BASIC tutoring text to generate FORTRAN turoring text

PERSONAL ASSISTANT AGENT FOR STUDENTS (PAA-S)

The PAA-S consists of three parts; student model, student behavior base, and tutoring module.

Student Model

The student model is used to assess the student's knowledge state and to make hypotheses about his/her conceptions and reasoning strategies employed to achieve the current knowledge state. Because most ICAI systems represent the student's knowledge state as a subset of an expert's knowledge base, the model is constructed by comparing the student's performance to the computer-based expert's behavior on the same task. This technique is named the "overlay model." Most early systems (e.g., SCHOLAR, Carbonell, 1970; WEST, Burton & Brown, 1979; WUSOR, Goldstein, 1982) used this approach. There are other techniques for computing the student model, e.g. analytic or transformational technique and synthetic technique. Analytic technique uses the background knowledge to transform the student behavior to the problem given, or vice versa or to verify if the student behavior and the problem given are equivalent or not. The specific operators needed to perform the transformation make up a student model. The synthetic technique involves obtaining a set of behaviors and computing a generalization of these by, for example, synthesizing elements from the background knowledge or input data (Raymund & Masamichi, 1998). Saeki (1999) presents his opinion about the new system called CSCL, which supposes that the goal of the student is not the same for the teacher's goal in all cases. However, we select "overlay" technique, since our domain knowledge is well defined, and our target students have the same goal of their teachers.

Student Behavior

The term "student behavior" has been used to refer to a student's observable response to a particular stimulus in a given domain. We store the knowledge state of mind of the student as well as his/her behavior in a database for future evaluation. Basic data items to be recorded are student name, class number, list of commands he/she practiced, number of correct answers, number of total questions, way of presenting he/she liked, and so

forth. The ratio of correct answers to total questions represents the effectiveness of the student's learning activities. This ratio with the way of presentation he/she liked are used to give a good hypothesis about the next presentation strategies and questions level of difficulties. Along with what information was presented, a record is also kept of which pedagogical strategy was employed for each of the information elements. This allows the system to infer the optimal teaching strategies for the student since the system has three teaching styles, text, Q&A, and dialog. One or more experts can generate the contents of these styles.

Linear Analysis

Our agent invokes the student model to check the student answer; it performs linear analysis, in which the stream of characters making up the answer of the student is read and grouped into tokens that are sequences of characters having a collective meaning, for example, the characters in the assignment statement:

```
let salary = 200
```

would be grouped according to our production rules, into the following tokens:

1. the reserved keyword "LET"
2. the identifier "salary" (variable)
3. the assignment symbol "=" (operator)
4. the number "200" (value).

The blanks separating the characters of these tokens would normally be eliminated.

During linear analysis, the agent spell checks for the reserved word character by character. For instance, LET is the reserved word of BASIC, when the student enters "la," the agent underline character "a" and an error message appears, guiding the student to correct it to be character "e." After the agent checks spelling of the reserved word LET, it groups the statement into tokens, and then the semantic analysis is invoked.

Semantic Analysis

After the linear analysis phase, the agent invokes the semantic analysis phase to check the student's answer for semantic errors. An important component of semantic analysis is the structure matching. Here, the agent checks each statement written by the student if it is permitted by the semantic rules specification or not.

Tutorial Module

A tutorial module is a set of specifications of what instructional material the system should present and how and when it should present it. In current ICAI systems, instructional strategies for specifying the presentation of learning materials are basically represented by two methods: the Socratic method and the Coaching method.

The Socratic method provides students with questions guiding them through the process of debugging their own misconceptions (Carbonell, 1970; Stevens, Collins, & Goldin, 1979). In the debugging process, students are assumed to reason about what they know and do not know and thereby to modify their conceptions. While the Coaching method interacts with the student according to the progress being made. In our system, the second approach was used, since it was easier for students who have not had enough background to answer the questions in the Socratic method (Osman & El-Khouly, 1993).

The system keeps the contents of the tutoring module in directories, each directory represents a name of one of computer's language under investigation, for example, "BASIC" directory contains the tutoring dialog's files of the BASIC language (see Figure 4). Each file name represents the tutoring dialog of one of the commands of that language. For example, "IF.TXT" contains the tutoring dialog for the command named "IF" (see Figure 5). This way gives us an opportunity to use the mother tongue of the student to represent the tutoring dialog of each command, since the tutoring text had been separated from the tutorial module.

Using the Tutorial Module

When the student accesses the module, the system asks about the computer's language, which the student wants to learn. The system opens the directory of this language and waits for the student to select the command

he wants to practice, to retrieve the associated tutoring dialog's file(s). The system presents the text that describes the command, and asks the student if he understands it or not. If the reply is "no," the system converts to another style for presenting the command (e.g., Q & A or dialog). If the reply is "yes," the system asks the student to write an example statement to check it. The system can accept a free format text from the student, and then infer the student module to check this statement. If the statement is correct, the system increments two counters, one for the number of questions which had been asked to the student, and the other is the correct answer counter. Otherwise, only the first counter will be increased. After the student completes this step, the system re-presents the first menu to allow the student to select one of the followings: (a) select another command, (b) present the score, and (c) exit.

RESULTS AND DISCUSSION

The teaching proceeds through seven phases during the academic year, with continuous evaluation and feedback throughout.

Phase 1. Introductory. One-month (2 hours/week) lectures were held to introduce the computer components and how the student can use it. For example, open power, using a mouse to select items, close it, cover it, etc.) and to learn basic terminology.

Phase 2. Grouping. Groups contain four students selected and the teacher assigns one of the computer sets to each group.

Phase 3. Try by yourself. When one student from the group begins his turn, the other members of the group watch what he is doing, and if they have a comment they can tell it, with the correction from the teacher.

Phase 4. Discussion. The teacher discusses the contents of the class again at the end of each session. He repeats the structure of the command, meaning, error places, and so forth.

Phase 5. Test. At the end of each month a written exam is held to test the students performance.

Phase 6. Judging and gifts. Those students, who get the final mark, receive valuable gifts at the morning queue in front of all school's students.

Phase 7. Evaluation and feedback. In order to meet the needs of teachers and students, feedback from all students is continually monitored. ATCL has been used in one class 2 hours/week for a six month course, in both elementary and primary schools with 11- to 14-years old students, for a total of 300 students. At the end of the course 10% of students got 100% the final mark, while 40% got over 70%. Comparing these data with the data of previous year, when the course was taught by traditional method, we found the average performance of the class was increased by 25%. Also, the parents of the students who complained about difficulty of the course at the beginning of the academic year, most of them purchased a computer set for their children at the end of the course.

It was a surprise for us that the students at the first and second year of primary schools (12-13 years old) got better performances than the third year students (14 years old). We tried to find the cause, and we found that the mathematical course for first year students at the primary level had been changed from an ordinary mathematical course to a modern mathematical course, and the student now at the third year got a mathematical course in the old form.

CONCLUSION

In this paper, the ATCL system has been presented for teaching computer languages to students who haven't any knowledge about computer programming. ATCL consists of two sub-agents, (a) personal assistant agent for teachers and (b) personal assistant agent for students. They communicate with each other as client-server. This allows extending the features of the system to communicate with other agents to exchange semantic rules and tutoring text for different languages. The tutoring text and the tutoring module were separated, such that we can use the student's mother tongue in the tutoring text. In the presented system, Arabic tutoring dialog was used. In the student model, the system can accept the free format answer from the student.

References

Brown, J., & Burton, R. (1979). An investigation of computer coaching for informal learning activities, *International Journal of Man-Machine Studies, 11*, pp.5-24.

Carbonell, J.R. (1970). AI in CAI: An Artificial Intelligence Approach to Computer-assisted Instruction, *IEEE Transactions on Man-Machine Systems, 11*, pp.190-202.

Carr, B., & Goldstein, I. (1977*). Overlays: A theory of modeling for computer-aided Instruction,* AI Lab Memo 406. Cambridge, MA: Massachusetts Institute of Technology,

Conati, C., Gertner, A.S., VanLehn, K., & Druzdzel. M.J. (1997). *On-line student modeling for coached problem solving using Bayesian networks*, In A. Jameson, C. Paris, & C, Tasso (Eds.*), User Modeling: Proceedings of the Sixth International Conference, UM97.* Vienna, Springer Wien New York.

Forbus, K.D., & Kuehne, S.E. (1998). RoboTA: An agent colony architecture for supporting education, *In AGENTS'98. Proceeding of the 2nd International Conference on autonomous agents.*

Goldstein, I. (1982). The genetic graph: A representation for the evolution of procedural Knowledge. In D. Sleeman & L. Brown (Eds.), *Intelligent Tutoring System.* London: Academic Press.

Hartley J. R., & Tait K., (1986). Learner control and educational advice in computer based learning, The study stations concept, *Computers and Education, 10*(2), pp.259-265.

Haruki, U., (1994). Integrated intelligent programming environment for learning programming, *IEICE Trans. Information & Systems.*, E77-D, No. 1.

Rosbottom, J., & Moulin, C. (1998). Using intelligent agents to change the delivery of education, *ACM SIGCSE Bulletin,(30)3.*

Osman, H., & El-Khouly M., (1993). *Tutoring educationsSystem for BASIC language: 3rd Conference in computer between theory and application,* Alexandria, Egypt: Marina time Transportation Academy.

Raymund S., & Masamichi S. (1998). Student modeling and machine learning, *International Journal of Artificial Intelligence in Education, 9,* pp.128-158.

Saeki (1999). *The expectations and problems with CSCL,* Invited talk 25th CSCU meeting, Saitama University, Japan.

Stevens A.L., Collins A., & Goldin S., (1979). Misconceptions in student's understanding, *International Journal of Man-Machine Studies, 11,* pp.145-156.