



Expert tutoring system for teaching computer programming languages

M.M. El-Khouly*, B.H. Far, Z. Koono

Department of Information & Computer Sciences, Faculty of Engineering, Saitama University, Urawa-shi, Shimo-okubo 255, Saitama 338-8570, Japan

Abstract

This paper presents an Expert tutoring system (E-TCL) for teaching computer programming languages through WWW. In this version, many teachers can cooperate together to put the curriculum of one/more computer programming language(s). Their contributions may include: (a) add or modify the commands' structure that will be taught; (b) generate different tutoring dialogs for the same command; and (c) generate different tutoring styles. On the contrary, the students can access the system through WWW, select any language they want to learn as well as the style of presentation they prefer and they can exchange their experiences. A personal assistant agent for teachers (PAA-T), a personal assistant agent for students (PAA-S) with an adaptive interface, and tutoring agent (TA) has been built. The TA resides on the server side and communicates via HTTP and IIOP with both the PAA-T and PAA-S on the clients side. This structure allows customization of the PAA-T and PAA-S to the needs of the teachers and students, without putting extra burden on the server. In addition, this allows having many teacher agents attending to the needs of a single or multiple student agent(s). © 2000 Elsevier Science Ltd. All rights reserved.

Keywords: Expert tutoring system; Intelligent agent; Computer aided instruction

1. Introduction

El-Khouly, Far and Koono (1999), presented a stand-alone version for an agent-based tutoring system for teaching computer programming languages. In this paper, we extend the features of that system to include Internet-based, adaptive learning environment and multi-agent technologies.

With the WWW as an educational platform, it will be feasible for the students to access the multimedia courseware with general-purpose browsers. No special tools are required to start learning. For the courseware provider, it is not necessary to worry about the distribution and maintenance of the copies of the courseware but they just take care of the original on their server (Nakabayashi, Maruyama, Koike, Kato, Touhei & Fukuhara, 1997).

The proposed system consists of three agents representing a server–client relationship, tutoring agent (TA) as a “server”, personal assistant agent for teachers (PAA-T), and personal assistant agent for students (PAA-S) as “clients”. The PAA-S can communicate with the TA through the WWW to retrieve the tutoring dialog of the command(s) that a student wants to practice, and to access

the experiences of other students in the blackboard module. While the PAA-T communicates with the TA to add/modify semantic rules of computer programming languages and to check the correctness of the contents of the blackboard database (as shown in Fig. 1).

2. Personal assistant agent for teachers

The object of the PAA-T is to standardize the decomposition of the language under investigation (as shown in Fig. 2), such that the TA can deal with all languages in the same way. The PAA-T consists of three parts, expertise module, semantic rules base, and tutoring text base. The PAA-T helps the teacher to cope with the knowledge base of a computer programming 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 “semantic rules”.

2.1. Expertise module

The expertise module consists of domain knowledge that the teacher intends to teach to the student. Representative 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 of scripts-frames (Hartley &

* Corresponding author. Tel.: +81-48-858-3489; fax: +81-48-858-3716.

E-mail addresses: elkhouly@cit.ics.saitama-u.ac.jp (M.M. El-Khouly); far@cit.ics.saitama-u.ac.jp (B.H. Far); koono@cit.ics.saitama-u.ac.jp (Z. Koono).

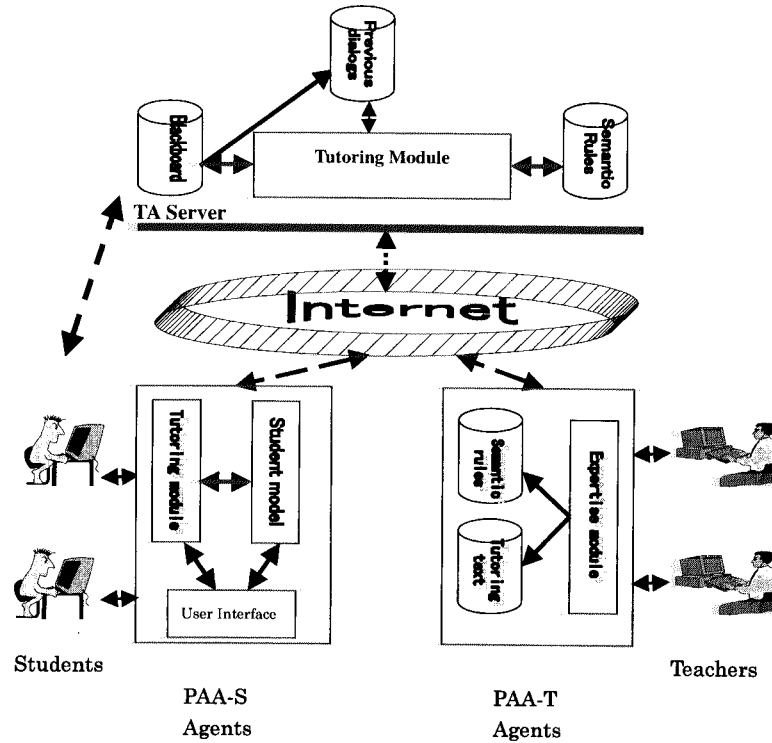


Fig. 1. E-TCL system.

Tait, 1986). In our system, a production rule to construct modular representations of commands is used.

2.2. Semantic rules

When a 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.

In this phase, the teacher puts the structure of each command (using upper case characters for reserved keywords only), e.g.

IF vov THEN s

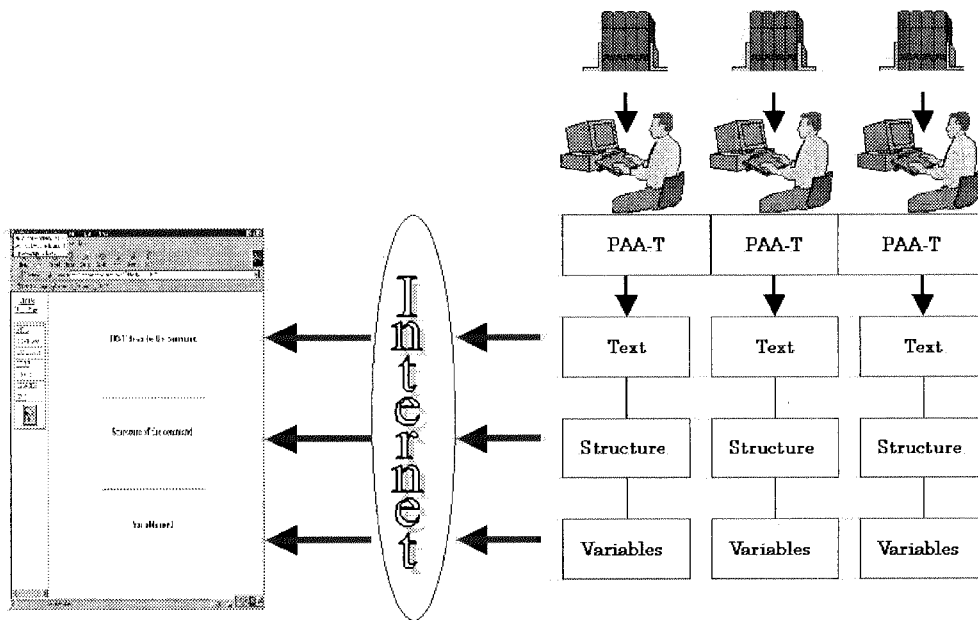


Fig. 2. Teachers decompose computer languages.

Means that one of the reserved keywords of BASIC language is called IF, and it can take the following structure (if the condition (variable operator variable) is satisfied then do the statements). Latter, PAA-S consults this semantic rules base to check the answers of the students (as we shall see in Section 3.1.1).

2.3. Tutoring text

The tutoring text-base contains the text that represents the commands of computer programming languages. Text is organized in terms of a conceptual network (Brusilovsky, Schwarz & Weber, 1996) hierarchically into lessons, sections, subsections, and terminal pages. Terminal pages contain the problems to be solved for the current command under investigation before it introduces new commands. Each teacher can contribute in this tutoring text-base: he/she provides an optimal learning path for an assumed average student. The tutoring text will be retrieved by the name of the command. At the database, the extension of the command's name represents the teacher's ID number as well as the order of this text. The ID number of the teachers and the order of the text will be used to determine which text should be present first at PAA-S, taking into consideration the teacher's specialist (since a text of one computer programming language can be used in teaching another computer programming language, as we shall see in Section 4).

2.4. PAA-T and TA interaction

When a teacher wants to send his/her contribution to the TA, the following data should be sent: a) teacher ID; b) computer language's name; c) command name; d) text describes the command; e) structure of the command; and f) variables used.

When the TA wants to consult a teacher about the correctness of students experience which is stored at the blackboard database, the following data should be sent: a) student ID; b) student behavior record; c) computer language; d) command name; e) problem description; and f) suggestion of solution.

3. Personal assistant agent for students

The PAA-S consists of three components, student model, tutoring module and user interface module.

3.1. 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 intelligent computer aided instruction (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". 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. We select the "overlay" technique, since our domain knowledge is well defined, and our target students have the same goal of their teachers. However, the system has the ability to support students who have another goal (i.e. not to complete the course, instead for example, they want to check the structures of some commands and compare it with other computer programming languages they already know). In this case, the system does not build student behavior records for them, and it allows them to jump through the commands without the required sequence.

3.1.1. Lexical phase

PAA-S invokes the student model to check the student answer. In order to accept free format answers from a student, one of the Compiler-phases had been used, it is called "Lexical Phase" (Lewis, Daniel, Rosenkrantz & Stearns, 1978). The lexical phase is concerned with breaking up the string of characters into the words they represent. For example, the answer of one student might be:

if salary > 200 then tax = salary * 0.1

The lexical phase would discern the fact that this character string represents the word "IF" followed by variable "salary" followed by the operator ">" followed by the number "200" followed by the word "THEN" followed by a statement tax = salary * 0.1 Again the last statement will be broken to be variable "tax" followed by an equality operator followed by variable "salary" followed by the multiply operator followed by the number "0.1".

Each token consists of two parts, a *class* part and a *value* part. The class part denotes that the token is in one of a finite set of classes and indicates the nature of the information included in the value part.

Moreover, during the lexical phase, PAA-S makes spell checking for the reserved word character by character. For instance, THEN is the reserved word of BASIC, when the student enters "tha", then the agent underlines the character "a" and an error message appears, guiding the student to correct it to be character "e".

3.2. 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 downloads the tutoring dialog of this language and waits for the student to select the command he/she 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/she understands it or not. If the reply is "no", the system presents another text for another teacher. If still the answer is "no" the system

converts to another style for presenting the command (e.g. Q&A or dialog) or consults the blackboard database. 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 model 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.

3.3. PAA-S and TA interface

Students can share their experiences through the blackboard module. The blackboard module has three components:

1. a global database called the blackboard (hosted at the TA-Server);
2. independent knowledge sources (from any PAA-S);
3. a scheduler to control knowledge sources and the blackboard database.

All experience elements are recorded in a structure, global database called the blackboard (Avelino & Dankel, 1993). The blackboard structure organizes experience elements along two dimensions, computer language name and command name. Each record contains the following fields (student ID, computer language, command name, problem description, and suggestion of solution).

When a student faces a problem, and he/she could not catch the required meaning from the teacher’s text, he sends a message to the message center at the blackboard module. A statement of the existing problem is displayed by the message center to all students on line. When a participant feels that he can contribute to the solutions, he sends his observations and/or conclusions to the message center. The scheduler organizes the knowledge source activity as well as the blackboard database and sends the contents from time to time to teachers to check its correctness, if it is correct it is moved to the previous dialog database (hosted at TA) (as shown in Fig. 1). Each teacher observes the messages written by others and considers those messages related to his specialty. Some of those teachers (who built the knowledge base for the same computer programming language under investigation) are able to offer immediate suggestions on what to do since their knowledge applies directly to the information currently in the message center. Others (who built the knowledge base for another computer programming languages), however, are forced to wait, possibly for an extended time, before their expertise is needed and can be applied.

3.4. PAA-S user interface

In most of the intelligent learning environments (ILE) systems, only the tutoring component is adaptive. The user interface usually looks the same for the novice and for the advanced student, while the student’s knowledge changes from the beginning to the end of a course (Brusilovsky, Specht & Weber, 1995). In PAA-S we use the student model for creating an adaptive interface. This is done in several ways: (1) last state adaptation; (2) visual adaptive annotation of links; and (3) function panel adaptation. By this way, the student can feel better, because when the interface is oriented towards an experienced student, the interface appears to be too complex for a novice student, and vice versa.

The browser downloads the HTML page and the applet code from the TA server to the PAA-S. The applet then runs on the client’s computer. First, the applet downloads the data file containing the initial variables about how the user interface should look like. Let us suppose that the data are stored in the file called *behavior.dat*, in the same directory as the HTML page. The applet then needs to open the URL, which is accomplished in the following JAVA code:

```
URL url = new URL (getDocumentBase(), “behavior.dat”);
```

This file, “*behavior.dat*” dynamically changes according to the student’s performance, and every time the above construct invokes, the suitable interface will be downloaded. There are restrictions, of course. JAVA applets cannot read or write from the disk on the machine that is running without the student’s permission.

3.4.1. Last state adaptation

Last state adaptation means adaptation to the last state of the user–system interaction, i.e. the system keeps the “settings” of the individual user (working directories, window position, etc.) and comes up when starting like the last time the user worked with it.

3.4.2. Visual adaptive annotation of links

PAA-S uses an extension of the traffic lights metaphor to annotate links visually. When presenting the current command tutoring under investigation, links to the other commands were annotated corresponding to a simple traffic lights metaphor referring to the knowledge state of each student. A red font of the link indicated that the corresponding command or section was not ready to be learned because necessary prerequisites were not met. The prerequisite relationships between commands are not represented directly, but the system is able to compute them from part-or and is-a relationships using several heuristics. A green font of the link indicated that this command or section was ready and recommended to be learned and a yellow font indicated that this command should have been learned. If the student visits a tutoring part of a command and did not successfully solve

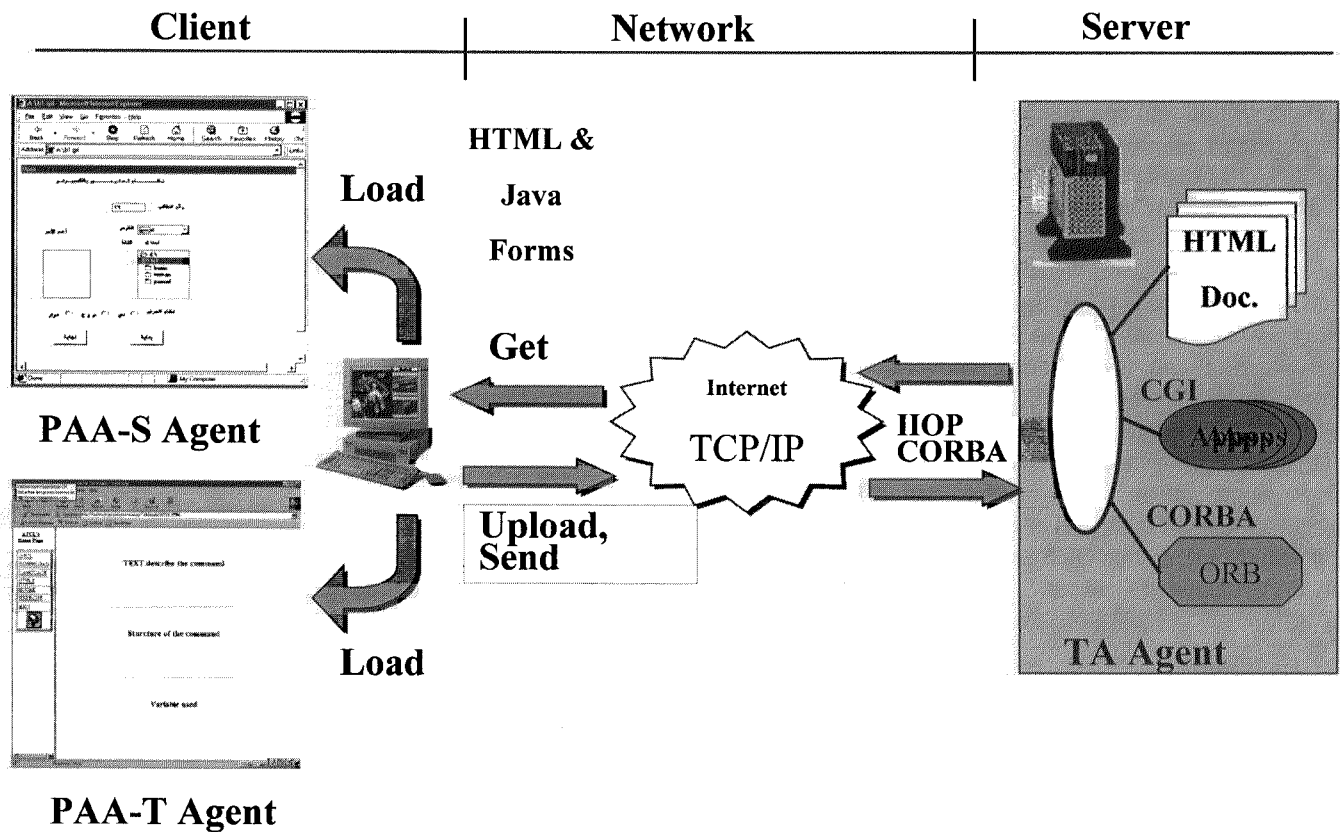


Fig. 3. The E-TCL using ORB.

the exercise, the link of this command will be marked as a green font not as a yellow one in the next invoke.

3.4.3. Function panel adaptation

Showing all functions in the function panel can cause working memory overload and be confusing to the novice student (Brusilovsky, 1993) suggested the use of a conceptual overlay student model to hide all unlearned constructs from an editor's menu. PSS-A used the following three techniques in combination.

1. *Hiding*: elements of the user interface, which the student is not ready yet to use.
2. *Dimming*: using the stable layout of menus regardless of the student's level of knowledge.
3. *Outlining*: using different colors for the functions buttons, to outline which functions were learned, and which were not.

4. Tutoring agent

TA contains tutoring modules, previous dialogs database, semantic rules base, and blackboard database.

There are two main features in intelligent tutoring systems (ITS) which are curriculum sequencing and interactive problem solving. These features differentiate

intelligent learning systems from computer assisted instruction (CAI) in that they incorporate intelligent techniques that skilled human teachers use in teaching classes (Gerhard & Sepcht, 1997).

Any computer language has command(s) for I/O, condition, loop etc. 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 TA can construct the tutoring module for the new language by retrieving the tutoring dialog of similar commands in another language and adapt it. This is very useful to produce several tutoring texts from different teaching points of view to fit students needs. For example, if the TA has the tutoring dialog of 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.

4.1. E-TCL and WWW interface

A main design consideration in the E-TCL version was to reuse as much of the component as possible. In the stand-alone version (El-Khouly et al., 1999), students receive immediate feedback on every action they take. If they enter an incorrect statement, the system will present a message error and guide them to the correct answer. HTML forms do not allow this kind of tightly coupled

interaction. The TA server receives information about student actions only when the student submits that information. In the E-TCL system, the TA, PAA-T and PAA-S agents can find and communicate with each other dynamically, using Common Object Request Broker Architecture (CORBA) as depicted in Fig. 3. CORBA allows agents to find each other and coordinate their behavior on a common object bus. This makes CORBA ideal for component-based applications. The advantages of using such technology in the E-TCL system is using those objects as a metaphor for using existing stand-alone ATCL applications as well as personalizing and task sharing between the client and server. In Internet based E-TCL, the JAVA ORB is used. With a JAVA ORB, an applet can invoke methods on CORBA objects using the IIOP protocol over the Internet. Consequently, there is no need to use HTTP and CGI programs that are the cause of extra overhead on the server, and also the client-side ORB enabled applets can be used in any JAVA enabled browser.

5. Conclusion

In this paper, the E-TCL system has been presented for teaching computer languages. E-TCL consists of three agents, TA, PAA-T and personal assistant agent for students. They communicate with each other as client-server through WWW. This allows the system to communicate with other agents to exchange semantic rules and tutoring text for different languages. The adaptive learning environment had been used in PAA-S. Moreover, the students can share their experiences through the blackboard system, i.e. the system can be dynamically enhanced by all members (teachers and students).

References

- Brusilovsky, P. (1993). In P. Brna, S. Ohlsson & H. Pain, *Student as user: towards an adaptive interface for an intelligent learning environment—Proceedings of AI-ED93, World Conference on Artificial Intelligence in Education*. Charlottesville, VA: AACE.
- Brusilovsky, P., Specht, M., & Weber, G. (1995). In F. Huber-Wäsche, H. Schauer & P. Widmayer, *Towards adaptive learning environments. GIS 95* (pp. 322–329).
- Brusilovsky, P., Schwarz, E., & Weber, G. (1996). In C. Frasson, G. Gauthier & A. Lesgold, *ELM-ART: an intelligent tutoring system on World Wide Web. Proceedings of the Third International Conference on Intelligent Tutoring Systems, ITS-96* (pp. 261–269). Berlin: Springer.
- El-Khouly, M., Far, B.H., & Koono, Z. (1999). Agent-based computer tutorial system—an experimental for teaching computer languages (ATCL). Special issue for intelligent agents for computer-based educational systems of the *Journal of Interactive Learning Research*. In press.
- Gonzalez, A. J., & Dankel, D. D. (1993). *The engineering of knowledge-based systems theory and practice*. Englewood Cliffs, NJ: Prentice-Hall.
- Hartley, J. R., & Tait, K. (1986). Learner control and educational advice in computer based learning: the study station concept. *Computers and Education*, 10 (2), 259–265.
- Lewis II, P. M., Rosenkrantz, D. J., & Stearns, R. E. (1978). *Compiler design theory*. Reading, MA: Addison-Wesley.
- Nakabayashi, K., Maruyama, M., Koike, Y., Kato, Y., Touhei, H., & Fukuhara, Y. (1997). Architecture of an intelligent tutoring system on the WWW. *Proceedings of the Eighth World Conference of the AIED Society, Kobe, Japan*.
- Saeki. (1999). The expectations and problems with CSCL: invited talk 25th CSCU meeting, Saitama University, Japan.
- Weber, G., & Specht, M. (1997). In A. Jameson, C. Paris & C. Tasso, *User modeling and adaptive navigation support in WWW-based tutoring systems. User Modeling: Proceedings of the Sixth International Conference, UM97*. New York: Springer.