

INTERNATIONAL CONFERENCE ON

M/SET 2000

**MATHEMATICS/SCIENCE
EDUCATION & TECHNOLOGY**

Proceedings of M/SET 2000–
February 5-8, 2000
San Diego, California

Editor
Robby Robson

Copyright © 2000 by the Association for the Advancement of Computing in Education (AACE)

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, without the prior written permission of the publisher.

The publisher is not responsible for the use which might be made of the information contained in this book.

Published by

Association for the Advancement of Computing in Education (AACE)

P.O. Box 2966

Charlottesville, VA 22902 USA

www.aace.org

Printed in the USA

ISBN 1-880094-38-X

Special thanks to—AACE Technical Coordinator: Jerry Price, Univ. of Houston.

Teaching Computer Programming Languages Through WWW

Mahmoud M. El-Khouly
Behrouz H. Far
Zenya Koono

Computer Sciences & Information Dept. Saitama University, Japan.
{elkhouly, far, koono}@cit.ics.saitama-u.ac.jp

Abstract: This paper presents web-based tutoring system (W-TCL) for teaching computer programming languages through WWW. In this version, two new features have been added: blackboard module and adaptive interface. With blackboard module a teacher can exchange his expertise with other teachers, and with adaptive interface the novice student will be satisfied because the system avoids complex interfaces. The system contains three sub-agents: the personal assistant agent for teacher (PAA-T), the personal assistant agent for student (PAA-S) and tutoring agent (TA). Using PAA-T, many teachers can cooperate together to: (a) put the curriculum of one/more computer programming language(s), (b) add or modify the commands' structure that will be taught, (c) generate different tutoring dialogs for the same command, and (d) generate different tutoring styles (e.g. text or Q&A).

Introduction

Learning one of the computer languages today, is essential for students in both undergraduate and graduate levels. However, the lack of good instructors is a problem. Using computer-aided instruction (CAI) and Internet technology can solve this problem.

The movement toward client-server applications began in the late 1980s and so in many organizations there are already many server applications with well-structured APIs for RPC or IPC access by a client. However, in many cases the backend server is a relatively standard database monitor, which provides no logic or protocol specific to the current application. The emergence of network computing, where the client side of the application logic is provided by JAVA applets that are downloaded at runtime to a Web browser, offers a new opportunity for constructing the agent-equivalent of a Web browser (Alper, 1997). 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 et al., 1997).

Lewis Johnson (Lewis Johnson, et al., 1997) support interaction with teachers and students through their project ADE, using off-the-shelf whiteboard and teleconferencing tools. But this required both teachers and students to be on-line.

The proposed system consists of three agents representing a server-clients relationship, tutoring agent (TA) as a "server", personal assistant agent for teachers (PAA-T), and personal assistant agent for students (PAA-S) as a "clients". The PAA-S can communicate with TA through 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 blackboard module. While the PAA-T communicates with 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 Figure 1).

Personal Assistant Agent for Teachers

The object of the personal assistant agent for teachers (PAA-T) is to standardize the decomposition of the language under investigation (as shown in figure 2), such that TA can deal with all languages with the same way. Each computer programming language has been stored at a different directory, and each file in its directory

represents a command name associated with (level of difficulty, text or Q/A or quasi). For example, the file named for01t.html at BASIC directory, means that this file contains the text describes the FOR command for first level. PAA-T consists of three parts, expertise module, semantic rules base, and tutoring text base. 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 it "semantic rules". To construct the semantic rules base, PAA-T uses BNF (for Backus-Naur Form) (Aho, etc., 1986).

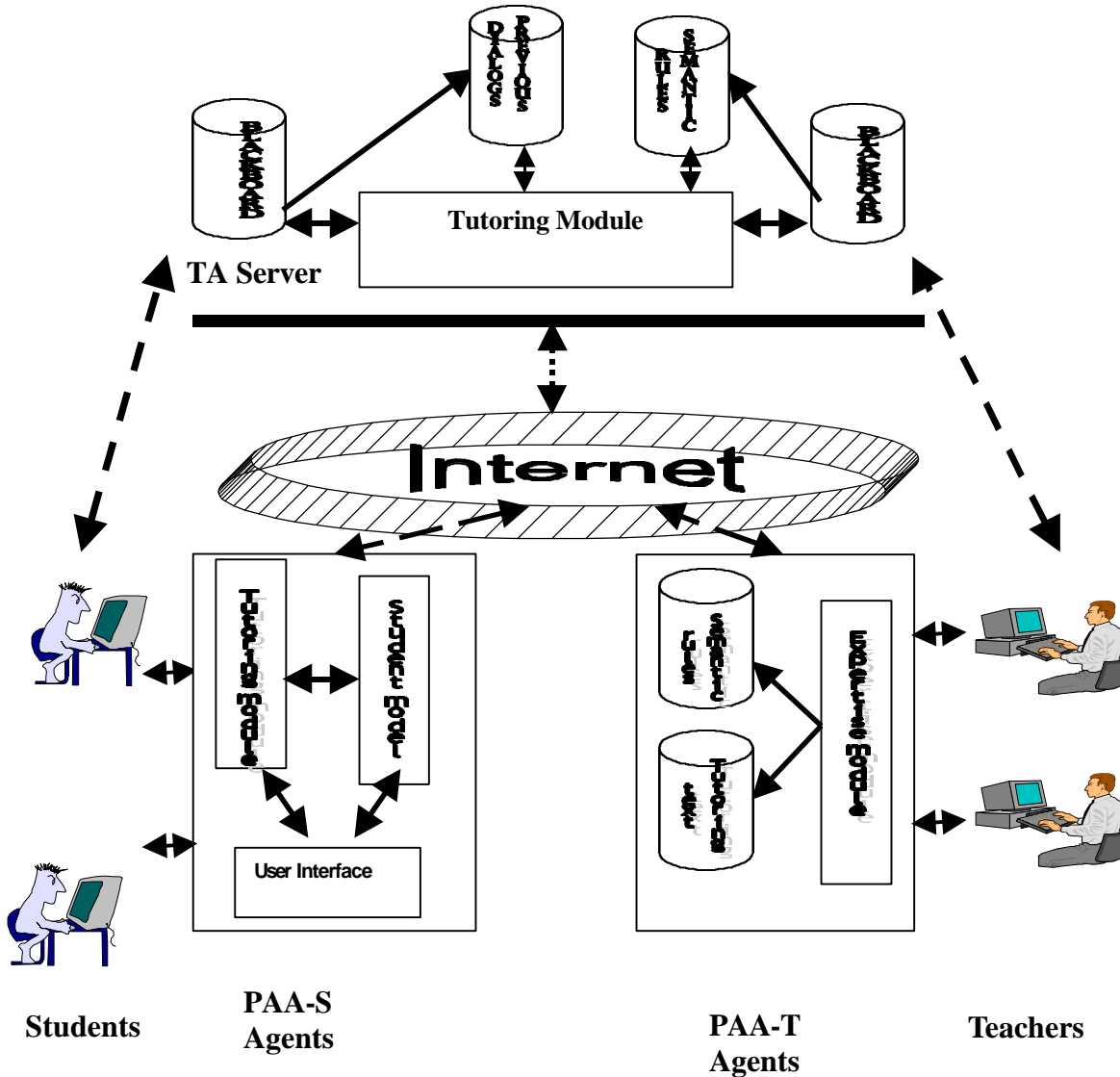


Figure (1): W-TCL System

Tutoring Text

Tutoring text-base contains the text that represents the commands of computer programming language. Text is organized in terms of a conceptual network (Brusilovsky, et al., 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 command. 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 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 later).

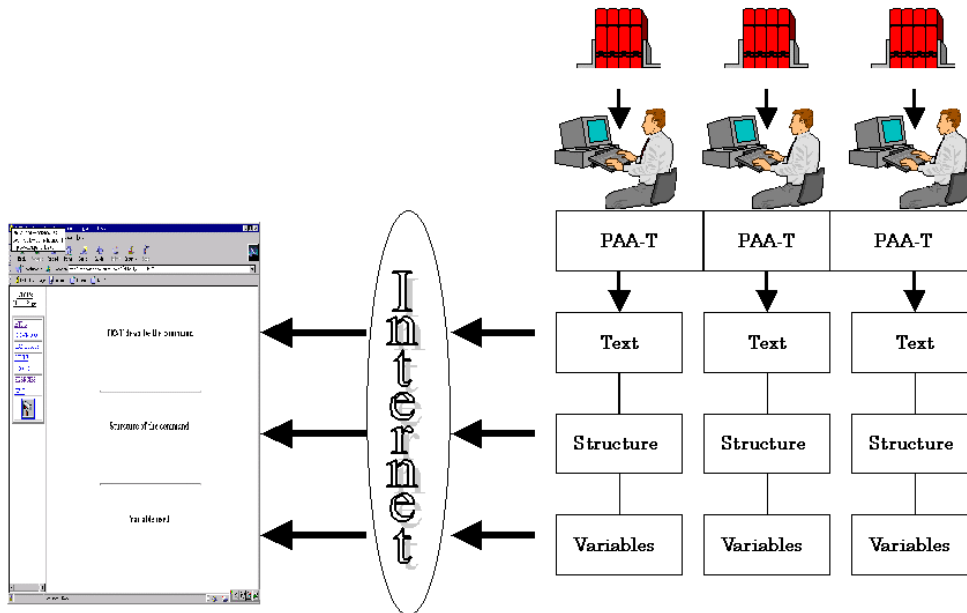


Figure (2) : Teachers decompose computer languages

Blackboard module

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

- global database called the blackboard (hosted at the TA-Server),
- independent knowledge sources (from any PAA-T),
- Scheduler to control knowledge sources and the blackboard database.

All experience elements are recorded in a structure, global database called the blackboard (Avelino et al., 1993). The blackboard structure organizes experience elements along two dimensions, computer language name and command name. Each record contains the following fields (teacher ID, computer language, command name, new quasi, and suggestion of solution).

When a teacher wants to generate some examples and he needs some previous examples even from different computer programming language, then he sends a message to the message center at the blackboard module. A statement of the example required is displayed by the message center to all teachers on line. When a participant feels that he can contribute to the solutions, he sends his contribution to the message center.

Personal Assistant Agent for Students

The personal assistant agent for students (PAA-S) consists of three components, student model, tutoring module and user interface module.

Student Model

PAA-S invokes the student model to check the student answer. In order to accept a free format answers from a student, one of the Compiler-phases had been used, it called "Lexical Phase" (Philip et al., 1978). The lexical phase is concerned with breaking up the string of characters into the words they represent. For example, if the system asks the student to construct a conditional statement that calculates the tax as 10% of the salary if the salary exceeds 200000 yen, then the answer of one student might be:

if salary>200000 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 "200000" 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 multiply operator followed by 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 lexical phase in the above example, PAA-S makes spell checking for the reserved word character by character. For instance, if the student enters "if salary > 200000 tha", then the agent will not accept any other characters and underlines character "a" and an error message appears, guiding the student to correct it to be character "e". Then the student can complete his answer. Note that, if the student enters "tex" instead of "tax", the system will not correct it, since the system checks only reserved words for the computer programming language under consideration. The system always consult *semantic rules base* to do that.

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 consult 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. If the correct answers of a student exceed 75% then the level of difficulty will increase by one, but if it is less than 50% the level of difficulty will decrease by one. 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.

PAA-S & TA interface

Students can share their experiences through blackboard module. All experience elements are recorded in a structure, global database called the blackboard. 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 content from time to time to teachers to check its correctness, if it is correct it is moved to previous dialog database (hosted at TA) (as shown in figure 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 suggestion 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.

PAA-S User Interface

In most of intelligent learning environments (ILE) systems, only 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, et al., 1995). In PAA-S we use the student model for creating an adaptive interface. This 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 verse.

The browser downloads the HTML page and the applet code from TA server to 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. Then according to the student performance, which will be recorded in the file, the browser downloads the suitable page for him.

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.

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 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 have been learned. If the student visit a tutoring part of a command and did not success to solve its exercise, the link of this command will be marked as a green font not as a yellow one in the next invoke.

Tutoring Agent

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

Any computer language has command(s) for I/O, condition, loop, and 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 tutoring agent can construct the tutoring module for the new language by retrieving the tutoring dialog of similar command in another language and adapts it. This is very useful to produce several tutoring texts from different teaching point of views to fit students needs. For example, if TA has the tutoring dialog of IF command in BASIC, then it consults the semantic rules of FORTRAN to substitute the BASIC command in tutoring dialog with the FORTRAN command.

W-TCL & WWW Interface

A main design consideration in W-TCL version was to reuse as much of 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 correct answer. HTML forms do not allow this kind of tightly coupled interaction. TA server receives information about student actions only when the student submits that information. In W-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). CORBA allows agents 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 W-TCL

system is using those objects as a metaphor for using existing stand-alone A-TCL application as well as personalizing and task sharing between the client and server. In Internet based W-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.

Conclusion

In this paper, W-TCL system had been presented for teaching computer languages. W-TCL consists of three agents, tutoring agent, personal assistant agent for teachers 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, both the teachers the students can share their experiences through the blackboard modules hosted at TA, i.e. the system can be dynamically enhance by all members (teachers & students).

References

- Alfred V.Aho, Ravi Sethi, and Jeffrey D. Ullman. (1986). *Compilers: Principles, Techniques and Tools*. Addison-Wesley Pub. Co.
- ALPER C and Colin H (1997) *Agent Source book: A Complete Guide to Desktop, Internet, and Intranet Agents*. John Wiley & Sons, USA.
- Avelino J. Gonzalez, & Douglas D. Dankel (1993). *The engineering of knowledge-based systems theory and practice*: Prentice-Hall, Inc. USA.
- Brusilovsky, P., Schwarz, E., & weber, G. (1996). ELM-ART: An intelligent tutoring system on World Wide web. In Frasson, C., Gauthier, G., and Lesgold, A., eds., *Proceedings of the Third International Conference on Intelligent Tutoring systems, ITS-96*. Berlin: Springer. 261-269.
- Hartley J. R., & Tait K. (1986). Learner control and educational advice in computer based learning: The study station concept. *Computers and Education*, Vol. 10, No. 2, 259-265.
- Kiyoshi Nakabayashi, Mina Maruyama, Yoshimasa Koike, Yasuhisa Kato, Hirofumi Touhei, & Yoshimi Fukuhara. (1997). Architecture of an Intelligent Tutoring System on the WWW. In Proc. 8th World Conference of the AIED Society, Kobe, Japan.
- Lewis Johnson & Erin Shaw (1997). Using Agents to Overcome Deficiencies in Web-Based Courseware. AI-ED'97 workshop on intelligent educational systems on the world wide web.
- Mahmoud El-Khouly, Behrouz H. Far, & Zenya Koono. (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*, To be published.
- Peter Brusilovsky, Marcus Specht, & Gerhard Weber. (1995). Towards Adaptive Learning Environments. In F. Huber-Wäsche, H. Schauer & P. Widmayer (Hrsg.): *GIS 95*, 322-329.
- Philip M. Lewis II, Daniel J. Rosenkrantz, & Richard E. Stearns (1978). *Compiler design theory*. Addison-Wesley Pub. Co.