EXPERIENCE USING THE ASA ALGORITHM TEACHING SYSTEM

Mário André Mayerhofer Guimarães
SENAC-DN, R.Gen.Justo 335/5 - 5 andar - 20021-130, Rio de Janeiro - RJ, BRAZIL.
mario@inf.puc-rio.br

Carlos José Pereira de Lucena
PUC-RJ, Rua Marquês de São Vicente 225 - 22453-900, Rio de Janeiro - RJ, BRAZIL.
lucena@inf.puc-rio.br

Maurício Roma Cavalcanti
SENAC-DN, R.Gen.Justo 335/5 - 5 andar - 20021-130, Rio de Janeiro - RJ, BRAZIL.

ABSTRACT

This paper describes the experience gained while using a computer based algorithm system called ASA, and explains why the evaluation motivated the need for high level structures to represent student and algorithmic knowledge. The system has been used in classrooms at SENAC, a Brazilian company associated with the Brazilian National Commerce Confederation, which promotes technical education. SENAC has a branch in every state of Brazil (26) and every year trains approximately one million students.

1) INTRODUCTION

The first section of the paper describes the available system. ASA consists of a tutorial with animations and an environment for program construction.

The next section describes the experience acquired from using ASA in the classroom environment. It describes its strong points and its limitations.

The third section focuses on ASA's enhancements (both implemented and forthcoming) that resulted from the evaluation of the software.

The last section focuses on ASA's present stage.

The description of the motivation for constructing ASA and the justification of the methodology it uses (ex: verbal protocols) as well as its comparison with other teaching systems, are not part of the present paper since they have been published elsewhere [Maye91-1, Maye91-2, Maye91-3, Maye92&Luce92].

2) ASA

ASA is presently a computer based environment for teaching algorithms at the introductory level. Its features include: Numeric Systems, Memory, Variables, Logic Operations, Sequential Operations (assignments, I/O), Selections, Iterations (While and For), Representations (Flowchart, Nasi-Schneiderman, Pseudo-code), Vectors, Sorting, and Matrices.

Computer based environments to teach algorithms fall into two basic categories [Maye92&Luce92]:

Animation of the program in execution, through data or code, to allow a better visualization of its properties.

High level programming, possibly with the use of diagrams, icons or forms. Sophisticated syntax editors and graphical interfaces are constructed for these purposes. These systems aim to free the student from details of language syntax, permitting greater concentration on algorithmic reasoning.

ASA includes both features mentioned above. The system originally consisted of three modules: Presentor, Animator and Constructor. However, the Presentor and the Animator have been merged into a single module called Lessons. It includes eleven chapters presented in tutorial form. The student is introduced to the subject through descriptive materials and animations of concepts and algorithms. Figure 1.1 displays one example of an animation. As the student clicks the button "Execute Next Command" the system executes one line of code and highlights the next line on the algorithm view located on the left of the screen. Simultaneously, the external and internal data views on the right of the screen are also updated. On Figure 1.2, the system prompts the student to type in the values of variables **a**, **b** and **c** after examining the code on the left side of the screen. For each answer, the system sends a message informing the student if it is correct or not and giving advice or a hint when appropriate. Furthermore, these answers are recorded on the students' card (profile) shown in Figure 1.3. Access to the card is granted only to the teacher.

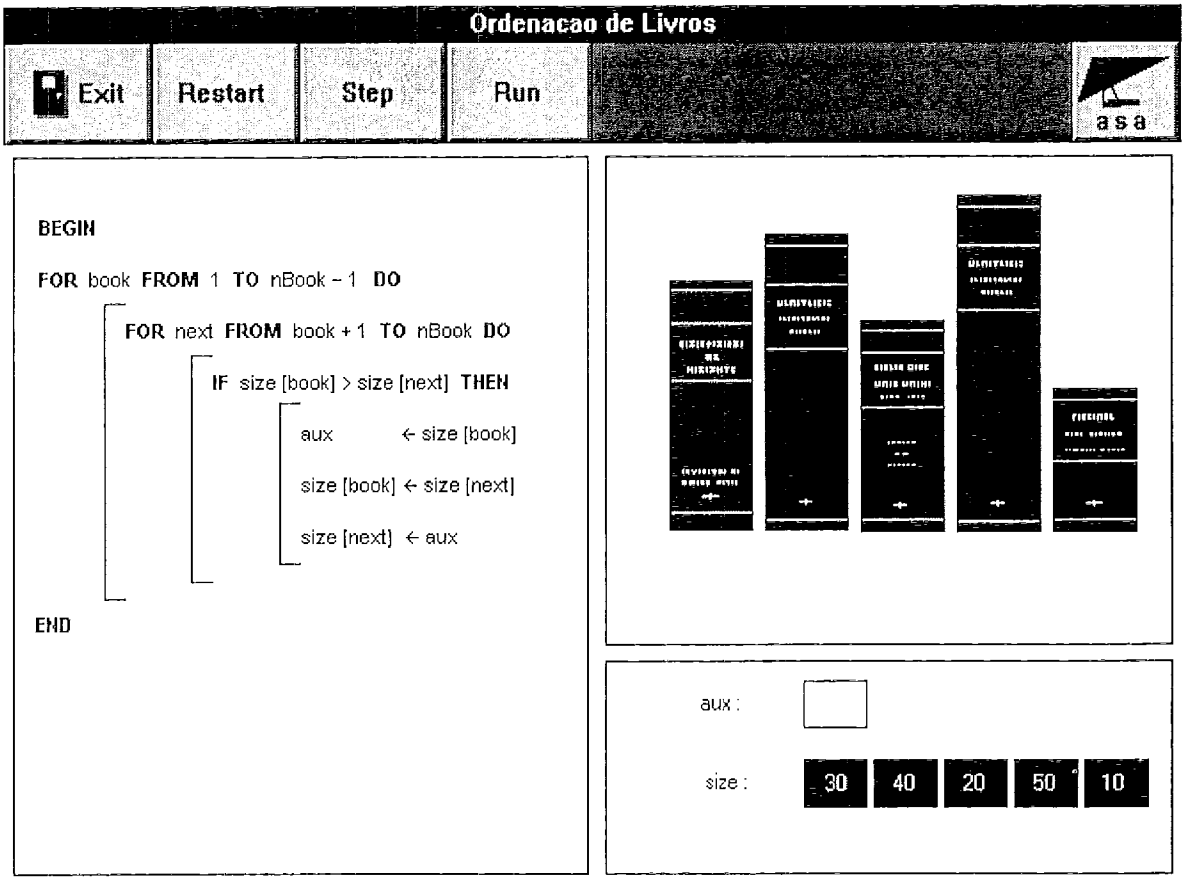

Figure 1.1

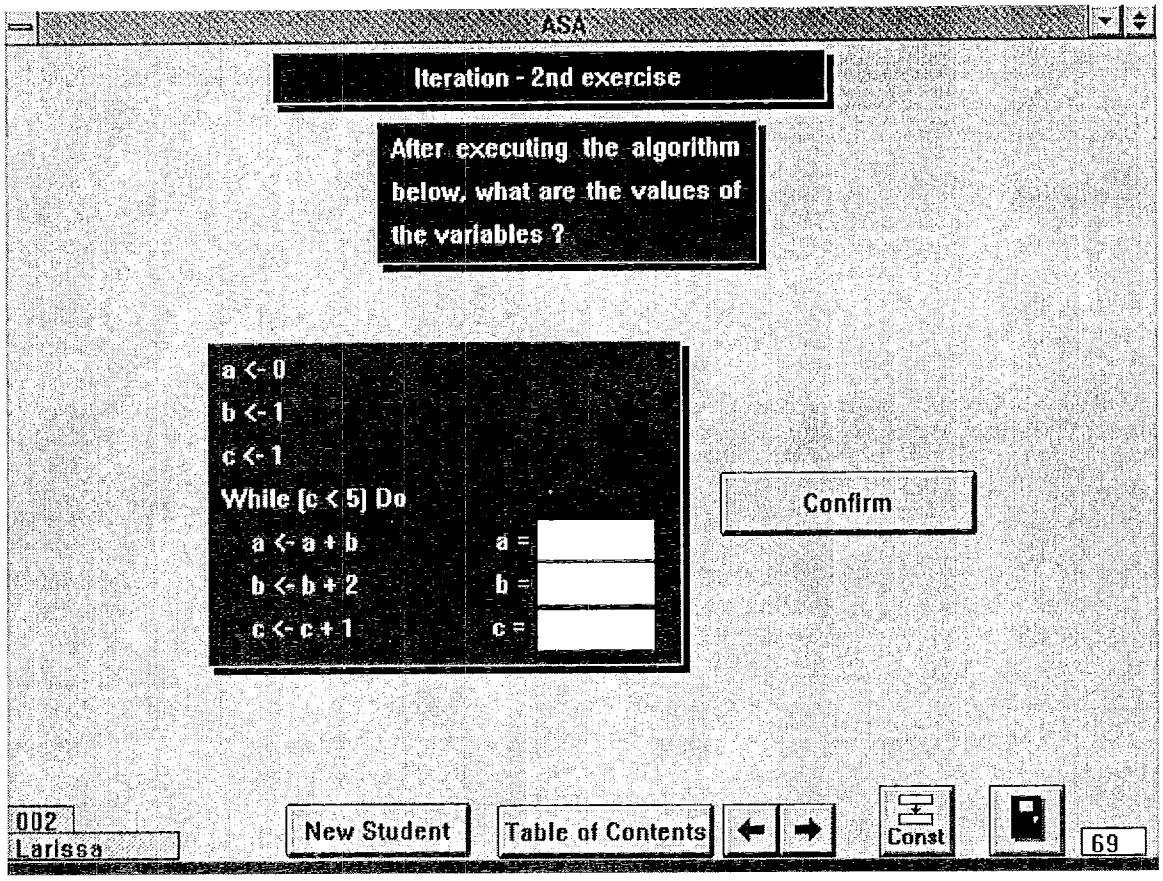

Figure 1.2

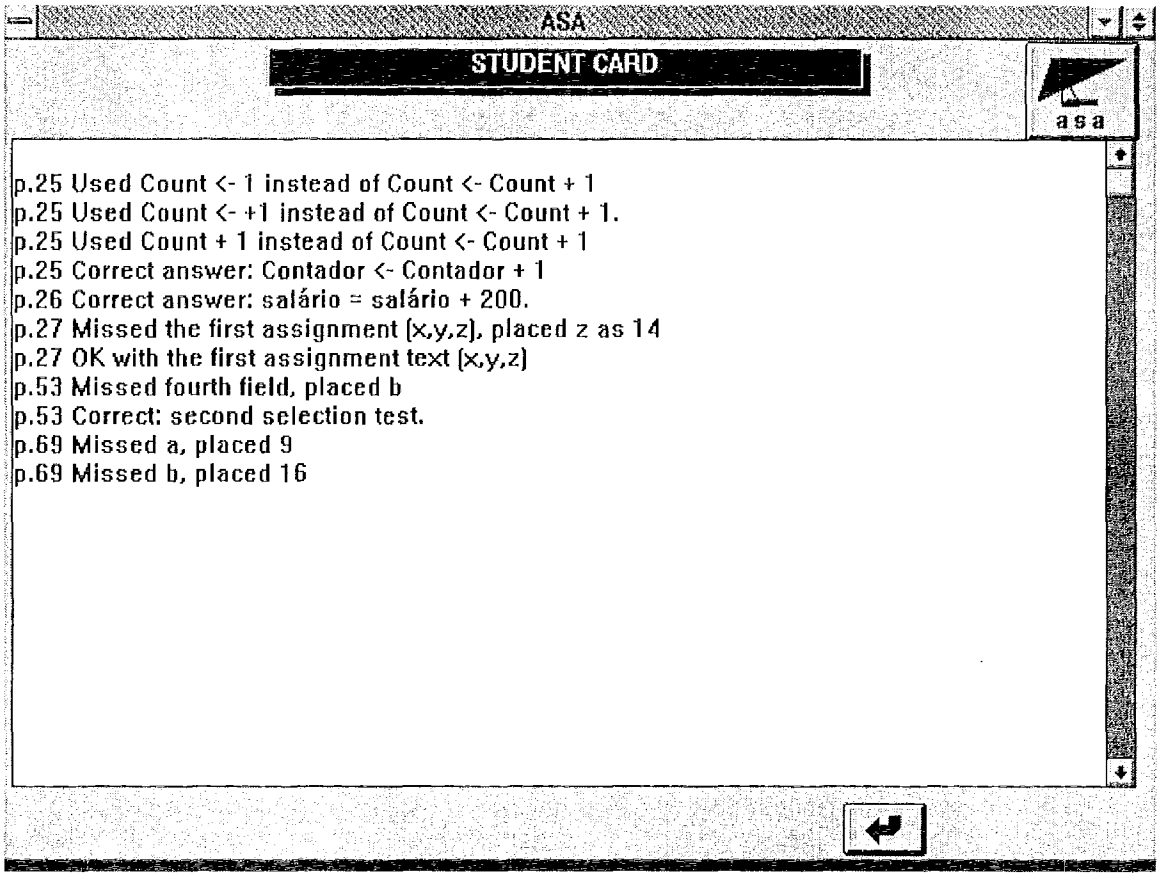

Figure 1.3

At the end of each lesson the system proposes a list of exercises to the student. Most of these exercises are to be done using the Constructor. The Constructor consists of a flowchart editor and interpreter. The student constructs his/her algorithms interactively using menus and icons, and the algorithm is represented graphically. The algorithms can be executed directly, step by step, or using breakpoints. An output window (displays strings and variables resulting from print commands) and a trace window (displays values of all variables) can be moved and sized to be placed at the most convenient location on the screen. The student can also visualize his/her algorithm in Pseudo-code, Pascal, C, or Clipper. Figure 2 displays a simple algorithm being constructed. Below the pop-up menus there are nine iconic buttons for creating algorithms (Variable Declarations, Assignments, Input, Output, If-Then-Else, While, For, Editing and Deletion) and three execution buttons (Execute Directly, Step by Step execution, and Abortion). The rest of the screen is the work area. In the example the student placed the trace window and the output window on the left side of the algorithm's flowchart representation and the Pascal and C windows on the right.

Every time the student makes changes and executes a new algorithm, the new version is recorded in the student's folder, thus allowing the teacher a closer follow-up on the classroom activities.

3) Using the Educational Software in the Classroom

The first time ASA was used in a classroom environment was in an "Introduotion to Algorithms" course. The conventional course was approximately 40 hours. With the software, the goal was to reduce it to 20 hours (it lasted approximately 25 hours). The first two class experiments involved only eight students (two per workstation).

3.1) Positive Observations

3.1.1) As with most educational software, one of the first noticeable advantages of using the system was motivation. With ASA, even the students who were learning at a slower rate enjoyed the classes. There was no need for homework or auxiliary material. This contrasts with a conventional course, in which students who are having difficulty learning a particular subject become unenthusiastic and tend to avoid the subject thereafter (ex: Math).

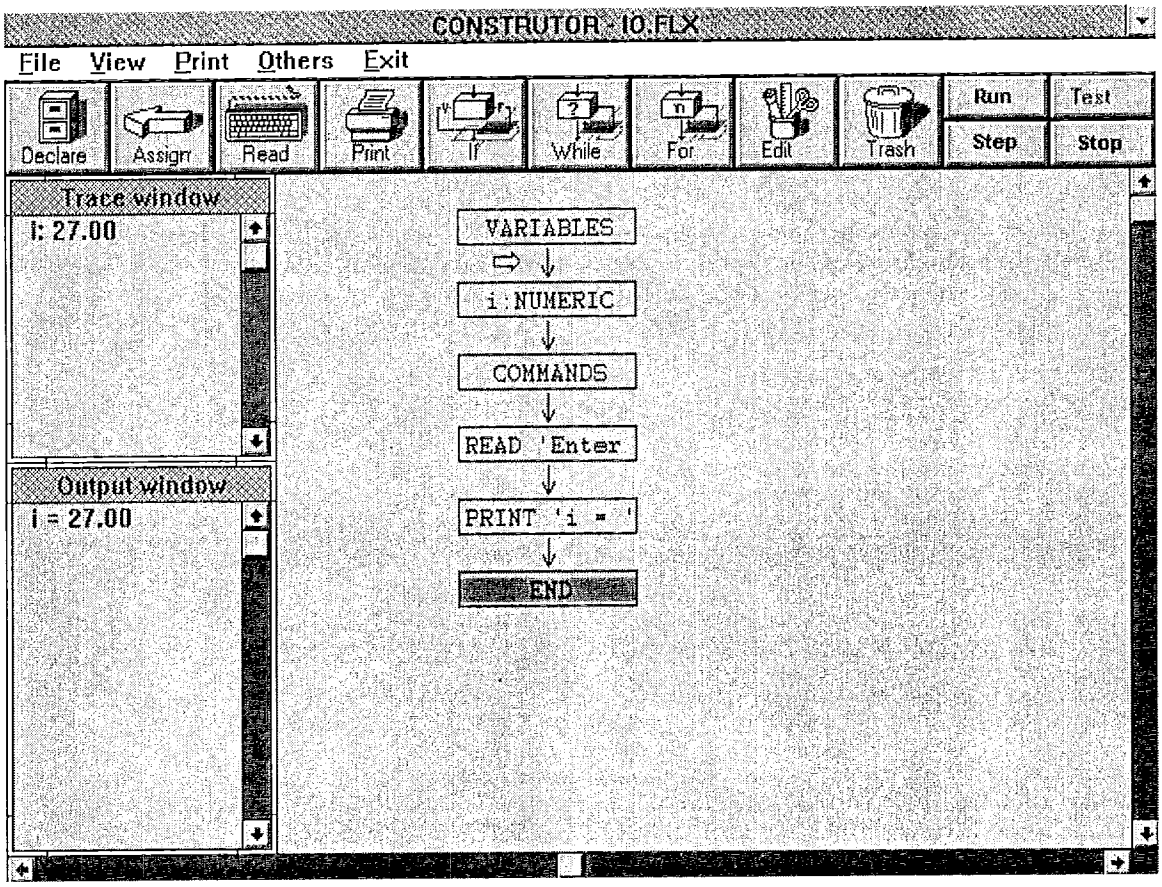

Figure 2

3.1.2) Another advantage of the system which is also common with educational software in general, is the possiblity of each student learning at his/her own speed. In conventional classroom environments, most students are either bored because the professor is reviewing the subject slowly, or are lost because the subject is taught too fast.

3.1.3) The software permitted students to map abstract into concrete views. In conventional courses, algorithms were either taught in an abstract form (the student never used the computer), or in a limited form (the student learned only one programming language).

3.2) ASA's limitations

Although ASA received much praise from educators and students, its first version showed two drawbacks, which reduced its usefulness for teaching introductory algorithms:

3.2.1) Many students were able to create algorithms correctly without understanding their purpose. It was necessary to explain that in a real world environment, the source code is not furnished to the end-user. This was done by showing the same algorithms running under DOS. Emphasis was put on such features as step by step execution and code visualization which do not exist in a real environment.

3.2.2) The teacher could lose control of the students during the lessons. It was impossible for the teacher to correct the students' algorithms and supply them with the necessary explanations/feedback (generating test data and displaying appropriate animation) as fast as they created new algorithms. During the first course, this problem was often resolved by the teacher intervening and furnishing information to the class at the blackboard. This helped the teacher to know what subject students were not understanding and to monitor them. However, this resulted in the loss of one of the biggest advantages that educational software offers: the possibility of each student learning at his/her own speed.

4) ENHANCEMENTS TO ASA

4.1) Evaluation of Students

For teachers to closely monitor students an educational software needs to capture their knowledge. The process was initiated by:

Making the Lessons more tutorial driven as opposed to the pure hypertext-like interface it used before. Questioning the student and recording their answers permitted the teacher to follow-up the students more closely in later courses (see Figure 1.3 of section 2).

Saving a version of the student's algorithm every time it is modified and executed.

Two more experimental courses were applied. The two enhancements mentioned above allowed for the teacher to check on the students' activities. Analyses of the students' files not only permitted the teachers to evaluate their students, but also supplied feedback to improve ASA's content. Whenever the same misconception occurred with more than one student, ASA was improved to give further information.

4.2) Program Verification

The next experiments were to be applied in larger classroom environments, consisting of approximately 20 students. Besides providing tools to evaluate the students, it was necessary to provide tools to relieve the teachers' burden. Therefore, the next enhancement built into ASA was the capacity to automatically verify the students' program. All the student needs to do is to click the button *Test*. ASA will execute the algorithm several times, one for each set of test data (that have been previously stored on disk for that specific algorithm). It will also inform the student if his algorithm is correct or not). If it is not correct, ASA will inform which set of test data caused the error.

4.3) Algorithm and Student Classification System.

Presently, ASA is capable of informing whether the students' algorithms are correct or not through the generation of test data. However, if the algorithm contains bugs, the system is not at present providing any useful information besides indicating the input data set that caused errors and showing what the correct output should have been. For a buggy algorithm, the system should provide complementary information indicating why it fails and the student's misconception.

Therefore, the project's next goal is to be able to classify both student and algorithm knowledge. Artificial Intelligence, concerning topics such as planning and knowledge based systems are being analyzed. Software Engineering topics such as program transformations are also being studied. These techniques, along with the student information described in section 4.1 that was received, has furnished input data for the Algorithm and Student Classification System that is being prototyped.

5) Final Remarks

The strategy adopted to achieve the above mentioned goals has been bottom-up, always accompanied by testing intermediate steps in classroom environments. ASA was applied in six experimental courses and distributed to the states of Brazil. Then, it was distributed to SENAC's branches. Results of applying ASA in different states have been reported. The discipline is now being taught in less than half the time. Besides, students that complete these courses have shown a better assimilation than students coming from conventional courses.

It is important to note that even though ASA's evolution is supported by theoretical concepts, it has also been motivated by practical considerations.

Reference

[ADAM80] Adam, A.; Laurent, J. "LAURA, A System to Debug Student Programs." Artificial Intelligence 15, 1980. p.75-122.

[BROW87] Brown, Marc. "Algorithm Animation", PhD Thesis, Brown University, May 1987.

[ERRI84] Erricsson, Anders ; Simon, Herbert. "PROTOCOL ANALYSIS, Verbal Reports as Data". The MIT Press. London, England, 1984.

[FOSD76] Fosdick, L.D.; Osterweil, L.J. "Data Flow Analysis in Software Reliability". Computing Surveys 8, 1976. p.305-330.

[GOEL89] Goel & Pirolli. "Design within Information-Processing Theory: The Design Problem Space". AI Magazine, Spring 1989.

[JONE90] Jones, A.C. "Teaching Programming at a Distance". Institute of Educational Technology, Summer 1990. p.130-133.

[JOHN86] Johnson, William L. "Intention-Based Diagnosis of Novice Programming Errors", Morgan Kaufmann Publishers Inc. Los Altos, California, 1986.

[KEAR87] Kearsley, Greg. "Artificial Intelligence and Instruction". Addison-Wesley Publishing Company. 1987.

[KRIS89] Krishnamoorthy, Kukkai; Swaminathan, Ramesh. "Program Tools for Algorithm Animation", Software Practice and Experience Vol.19 (6), June 1989. p.505-513.

[LUKE80] Lukey, F.J. "Understanding and Debugging Programs". International Journal of Man-Machine Studies, 12, 1980. p. 189-202.

[LURI70] Lúria, Alexander R. "Desenvimento Cognitivo". Editora São Paulo, 1970.

[MAYE91-1] Mayerhofer, Mario André. "Projeto de um Ambiemte para Animação e Simulação de Algoritmos (ASA)". Exame de Qualificação, PUC/RJ Dez/1991.

[MAYE91-2] Mayerhofer, Mário André. "Aplicações de Protocolos Verbais" Exame de Qualificação, Bruno Feijó. Dept. de Informática, PUC/RJ. 1991.

[MAYE91-3] Mayerhofer, Mário André. "Protocolos Verbais, Processos Cognitivos e Construção de Algoritmos". Boletim Técnico do SENAC. Vol.17. Número 2. Maio/Agosto 1991. p.107-124.

[MAYE92&LUCE92] Mayerhofer, Mário André; Lucena, Carlos José P. " Design of an Algorithm Simulation and Animation Environment", SIGCSE Bulletin, June 1992, Vol.24, n.12, p. 7-14.

[PAPE80] Papert, Seymour. "Mindstorms", Basic Books Inc., New York, NY., 1980.

[REIG89] Reigeluth, Charles M. "Instructional-Design Theories and Models: An Overview of their Current Status". Lawrence Erlbaum Associates, Inc. Hillsdale, New Jersey, 1989.

[RICH90] Rich, Charles; Waters, Richard C. "The Programmer's Apprentice". Addison Wesley Publishing Co., New York, N.Y., 1990.

[SCAN87] Scanlan, David. "Data Structures Student May Prefer to Learn Algorithms Using Graphical Methods", SIGCSE Bulletin, March 1987, p.302-307.

[SHEI81] Sheil, B.A. "Psychological Study of Programming". Computer Surveys, Vol.13, No.1, March 1981, p.101-120.

[SHU88] Shu, Nan C. "Visual Programming", Van Nostrand Reinhold Company Inc., New York, 1988.

[WENG87] Wenger, Etienne. "Artificial Intelligence and Tutoring Systems", Morgan Kauffman Publishers, Inc. Los Altos, California, 1987.

[WERT82] Wertz, H. "Stereotyped Program Debugging: an Aid for Novice Programmers." International Journal of Man-Machine Studies 16 (1982), p.379-392.

*****Teaching Continued From Page 44*****

Both techniques require consistent prodding by the instructor, particularly at the beginning of the term. If the instructor is not consistent in demanding the names, students will stop giving them. Getting back into the habit is then hard — better to not get out of it!

Which technique is better? It depends on the class. A highly-interactive class where instructors often question students, identifying each student with name cards is more suitable. A lecture oriented class where questions tend to originate with students is better suited for the first method.

Summary

Several teaching techniques have been discussed that are relatively inexpensive to implement, but, at least for one instructor, have yielded rich rewards.

Bibliography

[Bon91] Bonwell, Charles C. "The Enhanced Lecture" in *A Resource Book for Faculty*, Center for Teaching and Learning, Southeast Missouri State University, Sept. 1991.

[Ged92] Gedalos, Allan. *How to Make a Little Professor Go a Long Way.* Seminar presented at University of Waterloo on April 22, 1992. Gedalos is at University of Western Ontario, London, Ontario.