TECHNICAL REPORT

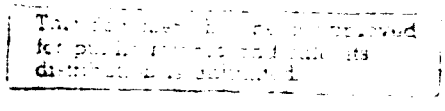
A Comprehensive Software Engineering Educational Experience

Norfolk State University

Department of Computer Science

Principal Investigator
George C. Harrison

ARPA Grant Number
MDA972-92-J-1024



A Comprehensive Software Engineering Educational Experience

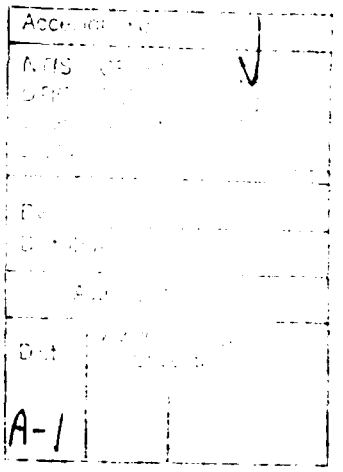
Norfolk State University

George C. Harrison
Principal Investigator
g_harrison@vger.nsu.edu

Department of Computer Science

Norfolk State University
2401 Corprew Avenue
Norfolk VA 23504

Grant Number MDA972-92-J-1024



TECHNICAL REPORT

August 15, 1993

THE ORIGINAL PLAN

The original plan of the project outlined in the proposal in August of 1991 was to experiment with possible solutions to the problems of offering a comprehensive software engineering course to undergraduates. The target student was assumed to be a junior computer science major with programming experience in Pascal through our CSC 260 (Computer Programming II) course. This course is approximately to the ACM designated CS2 course.

Some of the problems with software engineering education to undergraduates were identified to be

- The inability to complete quality documents: (user's manuals, requirements specifications, and detailed design) and implement the project during one 14 week semester.

- Since the ARPA call for proposals included an Ada education component, I felt that it was impossible to have students without an Ada background to complete the implementation of the software engineering project.

- The lack of undergraduate experience in the use of CASE tools.

The proposal, if funded, was to teach CSC 380 Software Engineering during the Spring Semester of 1992 and to complete the (Ada) implementation that summer. We were to

- Concentrate only on the interpretation and design of a software project with no emphasis on the language of the implementation.

- Design a project that was realistic and could useful upon implementation.

- Utilize on an irregular basis the expertise of technical writing teachers at the University.

- Produce a user's manual, a requirements document, and a detailed design document.

I was then going to teach our junior level course CSC 301 (Ada programming I) during the Summer of 1992. There would be no assumption that any of the students in the course had taken the CSC 380 course in the Spring nor any such course before. The goal of this course was have to students learn Ada via CAI instruction to be able to begin to implement the CSC 380 project in Ada by mid semester.

UPDATED PLAN

We were not notified of funding during the scheduled notification period: October 1991. Upon inquiries we found that there were colleges and universities that had been funded, but no notification of what schools were chosen was published until the late Spring of 1992. During the early Summer of 1992 we were notified that we were funded to do our project.

During the period between submitting the proposal and receiving notice of funding, the Computer Science Department was notified that it would have a network of 22 50MHz 486 Microcomputers by the Fall Semester of 1992.

Since the original proposal provided for several such microcomputers, we were able to rearrange our budget to provide a single 486 for the instructor to upgrade from an old 286 and to provide funding for licenses for 22 NSITE-Ada CAI packages, 22 Meridian Ada 386E Compilers, and 22 OpenSelect CASE tool packages.

The new plan was to

Offer CSC 380 (Software Engineering) during the Fall Semester of 1992. Have these students work on teams to provide a user's manual, a requirements document, and a detailed design document using the new PC's and the OpenSelect CASE tool.

Offer both CSC 301 and CSC 302 (Ada programming I and II) during the Spring Semester. The CSC 301 students were to learn Ada via the NSITE-Ada CAI package and their text book by the end March and then implement in Ada the CSC 380 project in April. The CSC 302 students, who were experienced Ada programmers were to build their Ada programming skills, become familiar with the special PC interfaces on Meridian's compiler, and then work on teams with the CSC 301 students to assist them in the implementation.

The CSC 301 students were to complete as much of NSITE as possible before March 19. The grade on the NSITE portion would be 50% of the final grade computed as (Lessons completed)/17 * 50%. (See below) They would also be graded on 4 programming assignments, a final examination (a take-home version in March), and their work on the implementation of the CSC 380 project.

THE STUMBLING BLOCKS

NETWORKING

By the end of the summer 1992, we realized that we would not be able to network our 486 computers until the middle of the Fall Semester. However, because of procurement problems we were not able to network the computers until March of 1993.

SOFTWARE PROCUREMENT

We also found that our procurement of software had to be placed on a request for bids. We were able to petition for a single source on some of the software. Because of the bidding problems and the confusion of the Vertix and Meridian merger, we were not able to receive the OpenSelect CASE tool until the middle of the Fall Semester when students were finishing their requirements document and preliminary user's manual.

OPENSELECT

The documentation for OpenSelect assumed knowledge and experience with CASE tools. The documentation also concentrated primarily on the information systems graphical tools, almost totally ignored the data dictionary, and had little information on data flow diagrams and Constintine charts.

Although the instructor had some CASE tool experience, it took almost two weeks to be able to teach the students how to use OpenSelect. It was the middle of November before students were able to work on the graphical tools. It was impossible to include data flow diagrams in the requirements document; we decided to place them in the systems portion of the detailed design document. Fortunately, Meridian is committed to redesigning and rewriting the documentation.

Since the computers were not networked, it was difficult for student teams to work on several computers while updating the same project.

NSITE-ADA

The biggest disappointment was with Network Solutions' NSITE-Ada. We found that we needed to install NSITE-Ada for each user. Meridian Software Systems, the reseller for Network Solutions' NSITE-Ada package, is only allowed to sell the single user version. Since NSITE-Ada, even without the inclusion of the Ada Language Reference Manual, expands to about 6 megabytes on a hard drive and since this problem was not noticed until the week before installation of the software and the beginning of the Spring 1993 semester, it was impossible to make any of the instructor's preferred modifications to questions and lessons. They would have to be made to each individual package. James Walker of Network Solutions was kind enough to develop a piece of software that allowed us to overcome the single user restriction and add up to five users for each installation of NSITE-Ada. Because of these delays we were not able to install the software and get students working on the CAI packages until the end of the second week of the Spring Semester.

Conceptually, NSITE-Ada is a good learning package. It allows the instructor to set up "Accounts" for students to learn all of Ada through 17 lessons. There are tests with a rather limited number of randomly chosen questions for each lesson. The instructor can set up the accounts to run through the lessons sequentially; that is, the student must pass the test at an instructor-defined level before moving onto the next lesson. I defined that level at 80% - the default value. The software allows the instructor to add questions, alter lessons, and add graphics and examples.

However, since each student is assigned a single account on a single computer, the instructor had to give accounts to students on each computer in a way that would attempt to avoid conflicts in the times of usage. We've found that NSITE uses some standards and word usage not used by the instructor. Also, if a student fails a test it is impossible for the student to review her wrong answers without the instructor's intervention in going to the instructor's account and reviewing the questions with the student. Some questions appear to be misleading or confusing.

It is impossible for the instructor to tailor the package for a student to install on her own computer without the intervention of the instructor. The student must have the account managers access to do this package and thus be able to see the answers to all the questions. The instructor was able to make up packages that just contained lessons and no tests so that students with home computers could read the material at their leisure.

MERIDIAN 386E Ada

This software caused the least problems. However, because we were not networked, the students had to keep their Meridian Ada Libraries on disks.

PC EXPERIENCES

Although most colleges and universities primarily use personal computers or workstations for most of their computer science education, Norfolk State students - especially juniors - have had little or no experience on these computers. Their experiences are primarily on VAX/VMS systems. This problem will no longer exist after 1993. Virtually all student work will be Unix, DOS, and Windows - based. Some of the students involved in this project had little personal computer experience.

SOME PROBLEMS

There were problems teaching CSC 380, but no more than the usual team problems and getting the students to work independently. Because of the learning problems with OpenSelect, the lateness of its arrival at Norfolk State, and the usual problems in getting students to work independently, some of their work with this software was minimal at best.

It is clear that students for many reasons are not learning Ada as well or as fast on NSITE as they would in a traditionally classroom setting. The instructor tried to assist these students by giving them an extensive set of notes used in previous Ada classes. Many students who have used these notes before had found them helpful. For the last two weeks in February and the class periods in March, the instructor had required students to come to class to review the lessons and former class notes with them.

SOME SUCCESSES

The instructor believes that the Software Engineering experience in CSC 380 was better than originally expected despite the usual team related problems and the problems with OpenSelect. The students did not have the pressure to be concerned about implementing the designed software, and for the first time in 6 years of teaching software engineering the documents were generally quite good. The best detailed design document was chosen to be used for the implementation of the project in CSC 301 and CSC 302.

Despite the problems with NSITE-Ada CAI software, some student are learned the basics of Ada very well on their own.

The final student implementations of the Software Engineering project were incomplete but clearly used Ada's features in an appropriate way. An complete implementation was finished during the Summer.

CONCLUSIONS

Like in a software engineering course where the students do much if not most of their work in teams and not in class, the Ada programming course had much of the same characteristics. There is always a feeling by the instructor that he doesn't have the control over immediate learning as in other classes. The instructor here had the students come to class more often that not to discuss their progress. The instructor also visited the computer laboratory several times during the day to help the students.

I believe that this system can work well - if we are to combine software engineering education with Ada education. However, it's apparent that all the software should be run off of a network to give students better access these packages. It's also apparent that there should be enough time for the instructor to design a student - oriented manual for the OpenSelect CASE tool. The instructor should be able to make the effort to modify the NSITE lessons to meet the objectives of the course.

- Students in the software engineering course produced documents of acceptable quality despite the problems.

- It is apparent from these experiences that the following is true:

- Use of a simple CASE tool can be helpful in the requirements and design processes. Students with enough classroom background on tool use and applications can benefit from the use of these software packages.

- Meridian's OpenSelect is an inexpensive tool for software development for undergraduates. Meridian is working on more complete documentation.

- Computer Aided Instruction (CAI) does not seem to work well in a paced but self-teaching method of learning a computer language. Instructor assistance is almost always necessary.

- NSITE's CAI single user package doesn't work well at all for regular instruction. The experiences in this course and in others show that NSITE works well for individual learning that does not include the testing with this package.

- Teaching juniors software engineering without an implementation brings almost superior results. Such courses can be language-independent. Students can concentrate more on the details of requirements and design and with writing quality user's manuals.

I feel that given enough lead time for acquisitions the successes on this project would have been significantly greater and the problems would have been minimized. The only real failure was using a CAI package as the primary way for undergraduates to learn a programming language.

Enclosed with the deliverables are the following documents:

1. A lecture outline for CSC 380 (Software Engineering)

2. A statement of need for the CSC 380 project.

3. A requirement document for that project.

4. A design document for that project.

5. The final source code for that document.