Ada: Still Our First Choice

Harold Youtzy, Jr.
Briar Cliff College

Abstract

Ada was first used at Briar Cliff in 1986. Its use continued and settled as the language of choice for the Data Structures course. An NSF grant was pursued and received for changing the Ada platform to 386s. The eventual success of that move has solidified Ada's place in our curriculum and may precipitate its move as the first language.

Historical perspective. In the spring of 1986, we fielded our first Ada course, after purchasing a Telesoft Ada compiler with Title III grant funds. The first course was an one hour mini-course, allowing myself and the students the opportunity to get acquainted with the syntax of the language and the operation of the compiler. Programming assignments were rudimentary and we considered basically the syntax differences between Ada and Pascal. The students' programming language experience to this point was fairly limited and given the "get-acquainted" objective of the course, we concentrated on converting rough draft program solutions in Pascal into more complete Ada programs. The objective of the course was met and we decided to continue our investigation of Ada the following year.

This time, we taught Ada as a full three hour course utilizing Cohen's text Ada as a Second Language[1]. Much of this course was still devoted to language syntax, but much more was covered than we were able to cover in the previous course. Indeed, we were able to cover Cohen's entire text, with the exception of the last chapter on low level programming. The response of the students was favorable, as it had been for the first course. The valuative distinction between the two classes was that more detailed programs were written in the second class. Also noted with the increased program size was that response time was less than encouraging. We were running our Telesoft Ada compiler on a VAX 11/750, and the students had been accustomed to the efficiency of the VAX Pascal compiler. Improvements were made in subsequent versions of the compiler, but our first version was quite slow. Still, we had set out to determine where Ada might fit within our computer science curriculum and this course had satisfied that objective. We concluded that Ada would be best introduced to our students within the Data Structures course. (Limited resources, such as textbooks, and compiler inefficiency precluded its use as the introductory programming language. Introducing it beyond Data Structures would limit its use to perhaps a single course). We then began teaching Data Structures using Ada as our programming language during the 1987/1988 school year.

Our initial failures and subsequent successes in that course for the first several years have been detailed elsewhere[2]. We agree wholeheartedly with Michael Feldman and others who claim allegiance to Ada in the Data Structures course. The topics addressed in this course match very well with the properties of Ada. The students within the course often become disgruntled over the fact that little time is spent on the syntax of Ada, but even so, they readily acknowledge the advantages Ada offers over Pascal. To help address their frustrations, we submitted an NSF Instrumentation and Laboratory Improvement Grant proposal.

NSF Grant. The impetus behind our grant proposal was to improve the student computer lab, which would hopefully better facilitate the use of Ada, not only within the Data Structures course, but in additional upper level computer science courses as well. As with many small, private liberal arts colleges, funding for improving hardware and software is, at best, limited. Indeed, it was with federal Title III Grant monies that our initial Telesoft Ada compiler was purchased. Additionally, we have a small number of majors in computer science, graduating five or less students each year, so courses which are taken solely by majors and minors in computer science usually have 5-10 students enrolled.

We have been teaching Pascal as the primary programming language in our entrance level courses in computer science. Turbo Pascal has been used for the last several years, and student performance has improved as a result. Turbo Pascal's operating environment has enhanced student interaction with the computer, reducing the number of errors associated with compiling and running programs, errors that are external to language syntax and programming logic. Or if not reducing these errors, then at least providing quicker solutions to them. Consequently, more attention could be devoted to developing principles for problem solving and less time was time needed with "administrivia," reciting, ad nauseam, all the "how-to's" associated with interaction of the student and the computer.

The inclusion of the above paragraph is not intended to provide an unconditional endorsement for Turbo Pascal, but rather to demonstrate that an user-friendly operating environment contributes substantially to the success of the student within the programming courses. This is particularly true when previous student use of a computer has been limited, and especially true if previous use included common software applications like word processing and spreadsheet. Most of our students fit within these categories.

Given the aforementioned student disgruntlement over spending little time dealing with the Ada language syntax, our objective became to improve the hardware and software in the computer lab. We felt that an improved operating environment could assist the student with learning the language syntax, or at least minimize the amount of time spent correcting compilation errors. Secondly, improved hardware and software could significantly improve turn-around time for the students. As the usage of the VAX by the faculty and student body increased, performance, in like fashion, decreased. Using Ada only worsened the performance. Although we were committed to using Ada, its advantages were being overshadowed by the symptoms common to a computer system that was quickly becoming outdated and overextended.

Our modest proposal, which was awarded, was to move Ada off the VAX onto 386-based machines. Eight machines were purchased for student use and two were purchased for faculty use. Additionally, a Micro VAX 3100 was purchased and serves as a network file server linking the grant purchased machines as well as other 386-based machines recently procured by the college. The Meridian Ada compiler was initially selected for the 386s to be run as stand alone, and the machines were later networked to the Micro VAX. Compiler upgrades were also installed as they became available.

Initial response. We would like to be able to inform our readers of the tremendous success we initially achieved as a result of the grant, but unfortunately, such was not the case. In the Data Structures course where we had hoped to have the most success, we have had the least success. When the 386s were purchased in the fall of 1990, the stand alone version of Meridian's compiler was installed. The class reaction was positive. Turn around time was dramatically improved. Although we felt the operating environment was not at the same ease-of-use level as that of Turbo Pascal, it was a significant improvement over what previously existed. The students took longer to adjust to the environment than they had with Turbo Pascal, but that may or may not be a reflection of the familiarity with that environment.

Our dissatisfaction with the system emanated from two specific instances. First, the students were using a floppy to store their programming files, and would frequently lose updates to their programs. An investigation yielded no explanation. Numerous cases were tried in an attempt to discover the reason for the loss. Different machines were tried in an attempt to isolate it as a hardware error, but our attempts were unsuccessful. The lost update files remain a mystery. Lack of familiarity with the software no doubt contributed to being unable to solve this mystery. Students were advised to save their work to the hard drive and then copy it over to their floppy. This obviously left the students with a poor impression of the software. This class of students, most of whom were computer science majors, had their first full taste of a software product whose delivery should have been delayed.

The second instance was compiler specific. A programming assignment was given to write a generic B-tree package, where the size of the B-tree would serve as the parameter to the generic package. However, when the package was instantiated, the value of the parameter was not available within the package. To be sure, the error may appear to be minimal, unless, as in our case, pronounced use of generic packages was to be made in the latter portion of the course.

Even so, we remained committed to Ada for the Data Structures course. We follow Feldman's text Data Structures with Ada[3]. The students are readily able to grasp the material in the text. The programming examples in the text are straight-forward and can be easily understood by those not familiar with the language (our students don't complain about not being able to read the language, just about writing it!!). The concept of packages fits very well with many of the principles taught in the course. Its use in later coursework is even more of an advantage, and all the more reason to include it as the language of choice for the Data Structures course.

In the spring of 1991, the system was used for our software engineering course. This was the second time Ada had been used in software engineering. The Software Engineering course is one of several upper-level courses that are offered on alternate years. The major improvement noted from our first try with Ada in Software Engineering was the turn-around time. The course is programming intensive and used exercises from the Software Engineering Institute the first time[4]. At the time this was taught, our VAX 11/750 was showing signs of being overtaxed and outdated. As a result, compilation was relegated to batch mode and given a lower priority. When we taught it for the second time using Ada, the students were glad to have exclusive use of a 386. A turn-around time of a minute or two was quite an enhancement from an hour or more. An in-house programming project was assigned the second time, and although neither class completely finished their respective assignment, our second effort allowed for more opportunity to pursue principles of software engineering.

In addition to running Ada on the 386s, two additional changes had occurred with our new system since the Data Structures course was taught the previous trimester. First, the 386 machines were placed on the network. Second, NSITE Ada, a tutorial for software engineering and Ada was installed.

Numerous difficulties with the networked machines ensued. The Meridian compiler required a substantial amount of base memory, which was less than what was available after the machines were networked. As a result, several machines were pulled off the network for use by the students in the software engineering class.

On a more positive note, the NSITE tutorial began to assist students who needed additional instruction on the syntax of Ada. Most, if not all of the software engineering concepts presented in the tutorial are also presented in the class. But the reinforcement of those concepts together with the language assistance was a major benefit for those in the class.

Again, the advantages offered by Ada for the Software Engineering course far outweigh those of Pascal. The program library maintenance facility helps to demonstrate to the students the effect of change to other compilation units.

Separate compilation, problem solving using packages, and re-use of generic packages are all advantages and sound arguments for using Ada in the Software Engineering course.

Second wave. In addition to purchasing the Micro VAX and the associated networking software, we have entered into Digital Equipment's Campus Wide Software License Grant Program. As a result, we have access now to the VAX Ada compiler. We have just completed teaching Data Structures utilizing the VAX Ada compiler. Other operating environment tools/resources are also available, but lack of time to become familiar with them precluded their use during the course. As such, we utilized only the language sensitive editor and compiler during the course.

Unfortunately, the level of skill present in the class just completed was significantly less than that of the previous class taught with the new hardware and software from the grant. The students, . with very few exceptions, seemed disinterested in Ada. Programming assignments frequently went undone. Given the small number of students in our class, the difference between an excellent class and a poor one may be the attitude or ability of just 3 or 4 students. It would no doubt have worked much better to have had the classes reversed. In our estimation, the first class could have adapted much easier to the VAX environment, and the second class would have been better off using the operating environment of the Meridian compiler. Consequently, in our effort to move to the VAX compiler and avoid the generic package parameter problems we previously had, we lost the advantage of having a known operating environment (students from the previous class were now unable to provide help, although students in the present class did little to seek help). Ironically, as a result of the poor skill level manifested by the class, we ended up not assigning programs utilizing generic package parameters, which was a primary reason for moving to the VAX Ada compiler!

The very first time we utilized Ada in the Data Structures course (1987/1988), the results were very discouraging, though no fault of the language. We had possibly one of our brightest classes of students (two valedictorians and a salutatorian were in the class). But poor preparation on our part and poor compiler performance on the VAX 11/750 combined to make it a less than memorable class. Now the shoes have been reversed, so to speak. A good compiler and sufficient preparation have led us to a group of students whose abilities are less than what we previously had. Perhaps this is their way of getting even for that first class!

Current trends. We are about to begin teaching our Operating Systems course utilizing Ada. It too, is taught on an alternate year basis. The previous time it was taught, students were given the freedom to select the language of their choice for their programming assignments. Although not all students chose Ada, our better students did. And although we expected better programs from the better students, we felt that the use of Ada allowed them to complete the assignments with less effort than the students in the class which opted not to use Ada. For the class presently before us, we plan to limit their language choice to Ada. We may, however, provide them with their choice of operating environments. We look forward to seeing what choices they'll make and the underlying rationale for them.

Into the future. We are very encouraged by the performance of our students in the just completed introductory programming course. We have seen both an increase in numbers and in student ability. The improved hardware has, no doubt, contributed to their fine efforts. Although the primary impetus behind the grant focused on the Data Structures course, we are pleased to see ancillary goals being met by the grant. We hope that the student performance and the hardware are the rain to end our drought.

We plan again to use Ada in the Data Structures course next year. Our goal is to make greater utilization of the resources at our disposal. The VAX Ada compiler has given us excellent performance. And it remains an advantage for the students to be exposed to different operating environments. Whereas we were, for a time, starting on the VAX using Pascal and then moving to PCs, our students will now be starting on the PCs utilizing Turbo Pascal and then moving to the VAX for Ada. But however they move, we feel that our best move has been to Ada. It remains our language of choice for the Data Structures course and beyond. And as teaching resources (i.e. textbooks) become available for Ada in the first programming course, we may give greater consideration to teaching Ada as our first language. That possibility was rather remote several years ago, but now it is at least on the table for consideration. The progress we yearned for six years ago when we first began teaching Ada, is now becoming reality.

Addendum. The operating system class alluded to in Current Trends above, proved to be very successful. Students chose three programming assignments from a selection of five options. Assignments were to be completed in Ada, but could be written on the Micro Vax or with 386s using Meridian's Ada compiler. The result came as a surprise to us. We anticipated favoritism of one environment over the other, but the result was favoritism of Ada over Pascal! Students did not seem partial to one system. But they did share their preference, however, for using Ada rather than Pascal. As the students had gained more familiarity with the language during the course of their studies, they recognized, to a greater extent, the versatility and potential of Ada as a programming tool.

We, too, have recognized that versatility and potential of Ada. Our use of Ada as the first language may only be a year away. The textbook resources are now available and the Ada platforms for PCs have had numerous improvements. Although the NSF grant was not the immediate panacea we hoped for, the long term effects have been very beneficial. Ada remains our first choice in the Data Structures course and may soon become our First language choice as well.

This material is based upon work supported by the National Science Foundation under Grant No. USE-9051068. The Government has certain rights in this material. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Norman H. Cohen, Ada as a Second Language. (New York: McGraw-Hill, Inc., 1986).

[2] Youtzy, Harold P. Jr. "A Healthy Marriage: Ada and Data Structures," Proceedings of the Fourth Annual ASEET Symposium, 13-15 June 1989, 59-64.

[3] Michael B. Feldman, Data Structures with Ada, (Reston, Va: Reston Publishing, 1985).

[4] Charles B. Engle, Gary Ford, and Tim Korson, Software Maintenance Exercises for a Software Engineering Project Course, Software Engineering Institute, Carnegie Mellon University, 1989.

Harold Youtzy, Jr. is an Assistant Professor of Computer Science at Briar Cliff College in Sioux City, located in northwest Iowa. He has been a member of the Math/Computer Science department since 1985.