MONITORING AN ADA SOFTWARE DEVELOPMENT PROJECT

Victor Basili
John Gannon
Elizabeth Katz
Marvin Zelkowitz

University of Maryland

John Bailey
Elizabeth Kruesi
Sylvia Sheppard

General Electric Company

This research program is monitored by the Office of Naval Research (ONR) under contract #N00014-82-K-0225 to the University of Maryland with funding from ONR and the Ada Joint Program Office.

As any science matures, the role of measurement, analysis and experimentation grows. The software engineering community has seen the continued development of new software development methods and tools and their use in various environments. The evaluation of methods and tools began with subjective criteria and has been developing toward more objective data collection, measurement and controlled experiments. While this trend is encouraging, these evaluation studies have largely been on a "one-shot" basis. What has been missing is a systematic approach which defines a long-range program for the study, analysis and evaluation of a specific method or tool.

The emergence of Ada provides a focal point for developing such a systematic study. As a first step, research teams from the University of Maryland and General Electric have embarked upon an eighteen-month collaborative effort. The purpose of this effort is to monitor the use of Ada on a realistically large and complex software development project within industry.

There are three areas concerning Ada which we felt could benefit from a study of Ada's use in industry. First, techniques for training and education in the new language will be addressed. The typical industry programmer is well-rooted in such languages as FORTRAN and assembler but may have little or no familiarity with many of the features and concepts underlying the proper use of Ada. Properly teaching the use of Ada to support software engineering concepts such as data abstraction and information hiding represents an important challenge. Second, we currently know little about the problems of designing and programming in Ada. For example, we would like to identify the types of errors which occur frequently with Ada and to determine whether programmers with different language or applications backgrounds use different features of Ada. Third, we would also like to identify metrics which are useful for evaluating and predicting the complexity, quality or cost of Ada programs.

The data collection and analysis for this study is being funded by the Office of Naval Research and the Ada Joint Program Office. The monitored software development is a part of General Electric's Internal Research and Development program.

The results of this project will be made available to the full community of Ada users. The purpose of this newletter is to invite our readership to criticize and to comment on our project, in the best of Ada's evolutionary style, so that it will result in the degree of benefit it was intended to provide. We therefore welcome feedback on our approach, design or goals. Please address your comments to either of the individuals listed below:

Dr. Victor Basili
Department of Computer Science
University of Maryland
College Park, MD 20742
(301) 454-4251

or

Dr. Elizabeth Kruesi
General Electric Company
1755 Jefferson Davis Highway
Arlington, VA 22202
(703) 979-6000

The Software Development Project

The Ada software development project that we are monitoring involves the redesign and implementation of a subset of an existing software system. This development was initiated as part of an IR&D effort to investigate Ada's suitability for a typical software project in General Electric's Space Systems Division. The development involves a portion of the ground support software for the Defense Satellite Communication System (DSCS-III). The original software was developed by GE and consists of approximately 100,000 lines of FORTRAN and assembly code. The selected subset originally required about 8,000 lines of FORTRAN. Among the functions to be re-implemented are an interactive process to receive an operator's inputs, graphics routines to display the output, and several concurrent processes which monitor telemetry data from the satellite.

Overall Project Goals

To ensure that the entire project is approached in a systematic and consistent manner, we formulated a set of goals for the project. All decisions about the data collection and analysis were evaluated with respect to these goals.

The primary question from which our goals are derived is "What can be learned from this empirical study that will aid in future Ada developments?" Goals were divided into two general areas: generic goals for any software project and Ada-specific goals for Ada as a design and implementation language. With each goal is associated several questions whose answers would help satisfy that goal.

The general areas, the constituent goals for each area, and questions for these goals are described in the Appendix.

Profile of Team Members

A four-member team has been selected to develop the software. The team includes a chief programmer, a backup programmer, a third programmer and a programmer/librarian.

Although a single, four-member project does not represent an experiment in any sense of the word, a great deal can be learned if the team is representative of software development teams in industry. The majority of experienced programmers in industry are not familiar with many of the features embodied in Ada. In selecting the members of the team, we were very interested in gaining some insights on the difficulties that are likely to be encountered by industry programmers. At the other extreme, we were interested in observing at least one programmer who is familiar with a wide variety of languages and concepts and, in particular, with Pascal. There is a general consensus that such a programmer should find learning Ada easier than one who only knows such languages as FORTRAN and assembly. Finally, we were interested in the experience of a novice programmer.

The development team was chosen to provide such a diversity of backgrounds. The chief programmer has substantial experience in the application area but has only FORTRAN and assembly language experience, while the backup programmer adds some exposure to COBOL, PL/I, Lisp, ALGOL and SNOBOL. The third programmer has just earned a B.S. in computer science and is fluent in Pascal and other block-structured languages. The librarian has essentially no previous experience except for a brief exposure to FORTRAN.

Training

Our goal in developing a training program for the software development team was to provide an in-depth and meaningful coverage of Ada which was within the bounds of what industry is likely to provide in the future. Thus, a full-time six-month course in Ada might produce better Ada programmers but would be unrealistic for an industry-sponsored training program. A total of one month was devoted to training.

The first four days were spent viewing the videotapes produced by Honeywell. These contain a series ot twenty-one lectures by Ichbiah, Firth, and Barnes and encompass a total of fifteen hours. The team members were permitted to stop the videotape at any time to ask questions or discuss points of interest.

The videotapes were followed by an in-plant seminar on Ada taught by George W. Cherry of Language Automation Associates. The seminar was given on six separate days spaced out over a four-week period. Between classes the team members reviewed their class notes, practiced compiling and executing programs on the NYU Ada/Ed interpreter, and worked together on a practice program, the final version of which was 500 lines long. The DoD Reference Manual for the Ada Programming Language and a number of articles of interest were also provided for the team.

To aid in creating a development environment which incorporates meaningful software engineering techniques and disciplines, the team was given a half-day class on methodology taught by Victor Basili of the University of Maryland. The class covered such topics as chief-programmer teams, design and code walkthroughs, program librarian and structured programming.

Project Monitoring and Data Collection

There were two criteria for selecting the software development techniques for this project: to establish a good development environment and to enhance our ability to monitor the development process. These techniques include the use of design and code walkthroughs to increase the visibility of error detection as well as the use of a project librarian to allow us to monitor the status of each component in the system. The librarian will enter all Ada PDL and code as well as all changes to the design or code.

A number of data collection forms are being completed by the programmers. These forms were designed to provide the information described in the goals (see Appendix).

We are also instrumenting the environment to collect a number of static measures and execution statistics. In addition, all source files will be saved for later analysis.

We would greatly appreciate your comments concerning the completeness of the goals and questions which are contained in the Appendix. Is there anything that you would like to know from this data collection effort which is not addressed by our current set of goals? Please address your comments to Vic Basili or Betsy Kruesi. Their full mailing addresses are given on the first page of this newsletter.

APPENDIX A: ADA GOALS

Area A: Generic goals for any software development project

Goal A1: Characterize the effort in the project.

1) How was the effort distributed over the phases of the project?
2) How was the effort for the project distributed over time?
3) How was the effort distributed across different functions in the software?
4) How are the error distributions similar to or different from other comparable software developments?

Goal A2: Characterize the changes.

1) How are the changes to the system distributed over the software development cycle?
2) How is the time for handling a change distributed? How long does it take to design and implement the change?
3) Is there a relationship between how far into development the change was needed and how much effort was spent on the change? How many sections it affected?
4 What kind of changes were made? (e.g., error correction, planned enhancement, etc.)
5) How many components are involved in the typical change?
6) How many changes are caused by a previous change?
7) How was the need for change determined?
8) How many and what kind of interface changes need to be made?

Area B: Goals relating to Ada as a design and implementation language

Goal B1: Characterize the errors made.

1) How were the errors found? (e.g., design review, inspection of output, etc.)
2) What were the non-Ada causes of the errors? (e.g., requirements misinterpreted, mistake in computation, etc.)
3) What features of Ada are commonly involved in errors?
4) Are there features of Ada that cause problems when they are used together?
5) Are errors attributed to confusion with another language? to a lack of understanding of Ada? to a lack of experience with a feature?
6) Are the errors made when using Ada as a design language different than those made when coding?
7) Where was the information found that was needed to correct the error? (e.g., Ada Reference Manual, another programmer, etc.)
8) Is the error characteristic of the feature or of the particular application it involved?

Goal B2: Determine whether certain aspects of Ada are difficult to use for certain applications.

1) Are there certain aspects of Ada that do not apply to this type of project?
2) Are there techniques usually used for this type of application that are difficult to implement in Ada?

Goal B3: Determine which aspects of Ada contribute positively to the design and programming environment.

1) Are errors easy to find? to correct?
2) Is there a large amount of parallel development once the interfaces are defined?
3) How effective is Ada in reducing interface errors? producing software that is easy to change? reducing the development effort, especially in realtime problems?

Goal B4: Determine which combinations of Ada's features are naturally used together.

1) How fully is the language used?
2) Are there certain features of Ada that are avoided because they are difficult to learn? difficult to use? poorly implemented? error prone?

Goal B5: Determine the effect of using Ada as a PDL.

1) Does Ada PDL allow sufficient abstraction at the early stages of design?
2) Is the language really being used as a design language?
3) Does the use of Ada PDL cause a preoccupation with syntax during the design stage?
4) What is the expansion of Ada PDL to code?
5) Does Ada PDL guide the design of the project or are portions of the system primarily other language programs written in Ada syntax?
6) Is there an adequate combination of features of Ada for use as a PDL?
7) Are the most expensive errors found while using a particular set of features of Ada as a PDL?
8) Are errors uncovered at the design stage that ordinarily would have been uncovered during coding because of the use of Ada PDL?
9) What percentage of the interface errors are uncovered during the design stage?

Goal B6: Characterize the programmers and associate their background with their use of Ada.

1) What are the programmers' opinions of Ada before they begin this project? during? afterward?
2) What is each programmer's background with other languages?

3) Are certain features of Ada or certain types of errors associated with particular programmers? Why?
4) Do certain programmers have problems with certain aspects of the language?
5) Do programmers want to use features available in other languages that are not available in Ada?
6) Are some features of the language overused, used incorrectly, or used inappropriately in the programmers' enthusiasm to use what they have learned?
7) Do people with no previous high level language experience have more or fewer problems with Ada than people with high level language experience?