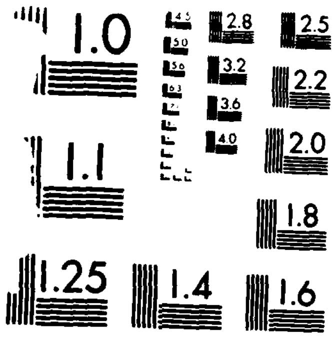
AD-A165 321 SOFTWARE METRICS FOR ADA(U) MARYLAND UNIV COLLEGE PARK 1/1
V R BASILI 03 FEB 86

UNCLASSIFIED F/G 9/2 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A

PROJECT FINAL REPORT

February 3, 1986

1. Contract

    Title: Software Metrics for Ada
    ONR Contract No.: N00014-82-K-0225
    Principal Investigator: Victor R. Basili
    ONR Scientific Officer: Dr. R. Grafton

2. Summary of Work

We have monitored a software development project written in Ada by integrating measurement into the software development process. Our goal is to identify areas of success and difficulty in learning and using Ada as a design and coding language. The underlying process and the evolving product are measured, and the resulting information characterizes this project's successes and failures. It may be used to make recommendations about training, methodology, and metrics to the Ada users community. This experience with data collection and metrics will aid in the selection of a general set of measures and measurement procedures for any software development project.

The project studied involved the redesign and reimplementation, at the General Electric Company, of a portion of a satellite ground control system originally written in FORTRAN. Four programmers were chosen for their diverse backgrounds and were given a month of training in Ada and software development methodology. They designed the project using an Ada-like PDL although a processor for the PDL was not available at that time. The design evolved into Ada code which was processed by the NYU Ada/Ed interpreter. The design and coding phases of the project extended from April 1982 to December 1982. Some unit testing of the project was done during the summer of 1983 using the ROLM compiler; however, the entire system has not been tested.

We used a goal-directed data collection approach from the beginning. Goals and objectives for the study were defined. Specific question and hypotheses were associated with each goal. Data collection forms and procedures were developed to address these questions. The forms and procedures were integrated into the software development methodology. The final step of this approach involved analyzing the data in order to answer the questions and either accept or reject the hypotheses.

Most recently, the data have been analyzed. All the data from the forms was entered in a database as were the data gathered by a processor which parses the design and code, checking for correct syntax and taking various measurements. Our conclusions are summarized below and elaborated upon in the technical report [Basili et al. 84] and the paper [Basili et al. 85] listed below. Further study into tools and metrics specific to Ada will continue in a following project.

3. Significant Results

Although the project studied ended part way through development, the results indicate what might happen in early stages of development in other projects. The data can be compared with the corresponding stages of other projects. The results from this project may prevent others from making costly management mistakes.

Learning Ada takes time. In this project it consumed 20% of the total effort. That time must be included in any estimate of effort for early projects using Ada. Training will probably have to be a continuing process as the team members learn the finer points of the language.

Ada is more than syntax and simple examples. The underlying software engineering concepts must be taught in conjunction with the support Ada provides for those concepts. Most programmers are not familiar with the methodologies developed in the seventies that Ada supports. Training in software engineering methodology and how to use it in the environment of a particular application is an absolute necessity for the proper use of Ada.

We do not know how Ada should be used. Ideally, our understanding of the software engineering concepts Ada supports would make the use of Ada natural. However, many people learn by example, and we do not have many good examples of how Ada should be used. We do not know how and when to use exceptions, tasks, and generics. We need to study various alternatives and show how they work with examples from various environment.

Design alternatives must be investigated. The design for this project was functional and more like than unlike the earlier FORTRAN design. A group at General Electric developed an object-oriented design for the same project. Neither of these approaches appears to be entirely appropriate. Just as a combination of top-down and bottom-up development is appropriate to many applications, a combination of functional and object-oriented design might well be most appropriate. Only after we know which type of design, or combination thereof, is best suited to the particular application can we teach people which design approach to use. Without such training, programmers will rely on their experience with other languages and will probably produce functional designs.

Proper tool support is mandatory. This project was done without a production-quality validated compiler. In addition to that very necessary tool, a language-oriented editor, which could have eliminated 60% of the observed errors, would have been desirable. This would have allowed the programmers to focus their attention on the logic errors that undoubtedly remain in the design and code. Data dictionaries, call structure and compilation dependency tools, cross references, and other means of obtaining multiple views of the system would have helped. A PDL processor with interface checks, definition and use relation lists, and various metrics would also be helpful.

Some methodology must be followed for a project to be successful. The methodology and tools to be used should be understood before the project begins.

The effect of the lack of good tools is mentioned above. In addition, the PDL was loosely defined until after design began. Effective design reading might have caught many of the errors. Even if we wanted to test this project after a compiler became available, we would have needed to create a test plan after the requirements were completed. However, that aspect of the methodology was deemed unimportant. The language is only one aspect of the environment and methodology. It cannot save a project in which the rest of the methodology is ignored.

We believe that this project is atypical in that it was done before a compiler was available and was not finished. However, it is typical in that training consumed an enormous amount of effort and the programmers were not familiar with the underlying software engineering concepts of Ada and that it might look like the beginning of many projects. The learning curve in methodology is quite large. As we study more projects that use Ada, we will learn how to teach it, how to use it, and where we might make mistakes. Until then, we need to study Ada and its use further.

4. Papers

J. Bailey, "Teaching Ada: A Comparison of Two Approaches," First Washington Symposium on Ada Acquisition Management, ACM, Laurel, MD, March 6, 1984.

J. Bailey, V. Basili, J. Gannon, E. Katz, E. Kruesi, S. Sheppard, and M. Zelkowitz, "Monitoring an Ada Software Development Project," *Ada Letters* II, 5, November 1982.

V. Basili, S. Chang, J. Gannon, E. Katz, C. Loggia Ramsey, N. Panlilio-Yap, M. Zelkowitz, J. Bailey, E. Kruesi, and S. Sheppard, "Monitoring an Ada Software Development Project," *Ada Letters,* Vol. IV, No. 2, September-October 1984.

V. Basili and E. Katz, "Metrics of Interest in an Ada Development," IEEE Workshop on Software Engineering Technology Transfer, Miami, Florida, April 1983, pp. 22-29.

V. Basili, N. Panlilio-Yap, C. Loggia Ramsey, S. Chang, E. Katz, "A Quantitative Analysis of a Software Development in Ada," Computer Science Technical Report, University of Maryland, 1984, UOM-1403.

V. Basili, E. Katz, N. Panlilio-Yap, C. Loggia Ramsey, S. Chang, "Characterization of a Software Development in Ada," *IEEE Computer,* Vol. 18, No. 9, Sept. 1985, pp. 53-65.

J. Gannon, E. Katz, and V. Basili, "Characterizing Ada Programs: Packages," The Measurement of Computer Software Performance, Los Alamos, New Mexico, August 1983.

J. Gannon, E. Katz, and V. Basili, "Metrics for Ada Packages: An Initial Study", in progress, 1984.

S. Sheppard, J. Bailey, and E. Kruesi, "Defining Metrics for Ada Software Development Projects," Proceedings of Computer Science and Statistics: 15th Symposium on the Interface, Houston, Texas, March 1983.