Comparing Ada and FORTRAN Lines of Code: Some Experimental Results

THOMAS P. FRAZIER
Institute for Defense Analyses, 1801 N. Beauregard St., Alexandria, VA 22311

JOHN W. BAILEY
Institute for Defense Analyses, 1801 N. Beauregard St., Alexandria, VA 22311

MELISSA L. CORSO
Institute for Defense Analyses, 1801 N. Beauregard St., Alexandria, VA 22311

Abstract. This paper presents the results of a study comparing pairs of functionally equivalent programs written in the FORTRAN and Ada languages. We found the Ada programs to require more lines of code than their functionally equivalent FORTRAN counterparts. However, we also observed that the overhead for Ada diminishes as program size increases. Our limited data suggested that there may be a cross-over point beyond which the size of an Ada program would be smaller than a functionally equivalent FORTRAN program. We explore some of the reasons for these economies of scale when using Ada. The implications of our findings on software cost estimating are also discussed.

Keywords: measurement theory, cost estimation, Ada, software size, size estimation

1. Introduction

A. Background

The introduction of Ada in the early 1980s has been cited as one of the major weapons in the fight to reduce the proliferation of computer languages and to control the cost of software in the DoD. The features of the Ada language were carefully chosen to enable good engineering practices and structure to be imposed during the development and maintenance of computer software. However, the use of these structural and engineering features presents new problems for the cost analysts responsible for estimating either the size or the cost of software systems to be developed in Ada.

Most software cost estimating models in use today assume that the cost of developing a computer program is a function of the size of the program plus other representations or measures of the complexity of the program, the skills and experience of the programmers, as well as other factors which affect cost. Typically, these cost models use lines of code as a representation of the size of a software program or project. For example, (Boehm, 1981) models the relationship between effort to develop software and the size of the code by using the following general form

\[ E = \alpha (KLOC)^\theta \prod_{i=1}^{n} m_i, \]

(1)
where \( E \) is defined to be the staff months of development effort, \( K LOC \) is defined as thousands of source lines of delivered code, \( \alpha \) and \( \beta \) are the parameters that define the baseline relationship between effort and size, and \( m_t \) are multipliers or cost drivers that account for differences in software product attributes, computer attributes, personnel attributes and project attributes.

The use of a line of code as a unit of measure is appropriate and effective when dealing with line-oriented languages such as FORTRAN or assembly languages. However, several problems arise when applying a FORTRAN-specific or line-oriented cost model to software being developed in Ada.

First, instead of being line-oriented, Ada is block-oriented, which means its statements and declarations can span several lines or be nested within one another. This implies that, instead of simply counting carriage returns, a special Ada-specific way of counting the effective number of lines in an Ada program is needed. Further, even given a way of measuring the size of an Ada program by some method for line counting, there is no assurance that a line of Ada by this definition will capture the same amount of function as a line of FORTRAN. This means that two functionally equivalent programs in the two languages might be considerably different in size, as measured by lines of code. Finally, there is no assurance that the development cost for a line of Ada by this definition will be the same as the cost to develop a line of FORTRAN.

**B. Objective**

This project addresses the functional size issues but not the programming effort issues raised when comparing the sizes of Ada and FORTRAN programs. The later question can be addressed by observing the cost required to develop Ada programs of various sizes. In theory, functional size issues could be handled by using function points to provide a language-independent measure of functionality (Jones, 1991). However, there are problems with using function points for this purpose. First, the ratio of function points to lines of code is assumed to be constant irrespective of the size of a program. We believe this may not be the case. Second, defining and counting function points is a matter of some debate and the focus of a great deal of current research. We believe our research may contribute understanding in both of these areas.

Information about the relative sizes of functionally equivalent programs is needed by any organization that is considering the use of Ada for application areas in which they have previous experience in FORTRAN. The reasoning is that such an organization would be able to estimate the size of a programming job if it were developed in FORTRAN. However, it would have no way of knowing whether an Ada solution would be more or fewer lines of code. What is needed is the added knowledge about how large an Ada solution to a problem will be, given an estimate of size for a FORTRAN solution. This knowledge will allow FORTRAN organizations to "bootstrap" their software cost estimating capabilities to include developments in the Ada language. Eventually, the need for this stop-gap technique will be eliminated by first-hand experience with Ada.

The focus of this study can be expressed in algebraic terms. The relationship between effort and size in line-oriented languages such as FORTRAN has been studied extensively.