A comparative case study of Programming Language Expansion Ratios

A thesis presented in partial fulfilment of
the requirements for the
degree of Master of Technology in Computing Technology
at
Massey University

Lim, Ping Hwee
February 1989

Abstract

An effective size estimation tool must allow an estimate to be obtained early enough to be useful. Some difficulties have been observed in using the traditional lines of code (LOC) measure in software sizing, much of which is due to the need for more detailed design information to be available before an accurate estimate can be achieved. This does not allow the result to be obtained early in the software development process. Moreover, the inherent language-dependency of LOC tends to restrict its use. An alternative measure using Function Point Analysis, developed by Albrecht, has been found to be an effective tool for sizing purposes and allows early sizing. However, the function point measure does not have a sufficient historical base of information for it to be used successfully in all cases with existing models of the software development process. Because lines of code already have a sense of "universality" as the *de facto* basic measure of software size, it can serve as a useful extension to function points. Language Expansion Ratios are seen as the key in providing such an extension by bridging the gap between function point and lines of code. Several sizing models have made use of expansion ratios in an effort to provide an equivalent size in lines of code in anticipation of its use in productivity studies and related cost models. However, its use has been associated with ranges of variability. The purpose of this thesis is to study ***Language Expansion Ratios***, and the factors affecting them, for several languages based on a standard case study.

This thesis surveys the prevailing issues of software size measurement and describes the role and importance of language expansion ratios. It presents the standard case study used and the methodology for the empirical study. The experimental results of measurements of the actual system are analysed and these form the basis for appropriate conclusions on the validity and applicability of the expansion ratios studied.

This research shows that the use of Language Expansion Ratios is valid but it is considered inadequate when applied in its present form. This was found to be due to the weighting factors associated with the appropriate function value obtained for the different functional categories of the system.

Acknowledgments

I wish to extend my gratitude to Professor Graham Tate for consenting to be my supervisor for this thesis. His direction has been very important to this work. I am also grateful for his careful review and comments of the entire content of this thesis. His guidance and encouragement are very much appreciated.

A special thanks to Miss June Verner for her constant help and support during the course of this research. In particular, I wish to express my deep appreciation for her help in making the documentation facilities available to me.

Finally, I would like to express my gratitude to Miss Lili Ton for her help during the preparation of this thesis. Her efforts and patience in some of the typescripts and drawing of the many diagrams deserve my heart-felt thanks.

TABLE OF CONTENTS

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

The impact of the Software Crisis emanates from the realization of the importance which computerization has on society and industries. An overwhelming demand for software escalated to such an extent that traditional software development techniques could not cope. The problem related to the difficulty of establishing techniques for handling the growing size and level of complexity of software systems whose development schedules could not be accurately predicted.

The cost of software grows disproportionately with the other associated costs of computer systems and this upward trend is an issue of concern [DACS87,JONE86]. In the search for a solution to the software crisis, the field of Software Engineering was created in a deliberate attempt to use a combination of techniques, methods, and tools for producing economical software that is reliable and works efficiently.

An important aspect of Software Engineering has been the focus on software size estimation as a prerequisite for resource planning and scheduling in the software development process. The unpredictable nature of the software development process has prompted intense research in finding a way of making programs measurable and, hence, more predictable. Source lines of code (LOC) have been widely used for research studies involving size estimation. The count of the number of lines of code is said to be related to the size of the effort required in the development process. Several studies have been pursued to verify this.

The study by Walston and Felix [WALS77] discusses research into a method of estimating programming productivity by measuring the rate of production of lines of code by project and relating them to factors which might influence its behaviour. Measurement data were collected from 60 projects in one organization and were maintained in a measurement database. Based on these data, productivity analysis of effort and product size shows a nearly first-power (or linear) relationship. In another study carried out in the Software Engineering Laboratory at the University of Maryland, Basili et al [BASI81] examined the relationships among the various basic software development variables, such as size, effort, project duration etc. The analysis reveals a high probability of a relationship between total effort and delivered lines of source code. The relationship is nearly linear with a coefficient of determination of 0.93 at the 0.001 level of significance. This high correlation indicates a possibility of using source lines of code to predict the total effort required in a development project. This result was found to be consistent with the study by Walston and Felix [WALS77].

The validity of using source lines of code as a predictor of programming effort is also based on the assumption that it includes a measure of functional complexity. Although studies by [WALS77] and [BASI81] indicate a linear relationship between source lines of code and effort, Basili et al [BASI81] also stated that the assumption which relates the functional complexity to program size is subjective. The argument is that size may increase at an even greater rate as complexity increases. This follows the notion suggested by Brooks [BROO75] that man and month are not necessarily interchangeable as the result of increasing one may not directly cause the other to decrease. However, it can be acknowledged that the link between lines of code and effort prediction is a valid one, though more factors relating to complexity will have to be considered. Already, several cost models have used lines of code as the input parameter for cost and schedule estimates. Of the better known costs models are the COCOMO and SLIM models [BOEH84].

The biggest difficulty in using today's Software Cost models has been the problem of providing sound sizing estimates. Several methods of size measurement have been developed to depart from the usual lines of code measure for size estimation in an attempt to search for a better way to estimate software size early enough for it to remain useful. One such method that remains popular is the measure for function value developed by Albrecht called Function Point Analysis. Function Point Analysis has been quite successful and is adopted in several commercial software sizing models. However, most of these models also provide an equivalent size in lines of code in anticipation of its use in productivity studies as well as with existing cost models. The use of an expansion ratio to convert from function points to lines of code has been recognized to contain ranges of variability due to the fact that it is obtained from different sources or based on different datasets. However, the expansion ratio remains useful in standardizing the basic unit of measure for quantifying software. The purpose of this thesis is to study ***Language Expansion Ratios***, and the factors affecting them, for several languages based on a standard case study.

Chapter 2 provides a summary of the background information relating to the issues of software sizing. A review of software metrics and the problems associated with them are presented followed by a discussion of the subjectivity of sizing in existing size estimation models. It emphasizes the need for an improved size estimation method which can be applied to the wider range of programming languages and examines the role of language expansion ratios in providing such an extension to existing sizing methods.

Chapter 3 presents the research objective for this thesis and a brief outline of the system analysis and design specification of the case study used. The languages used in this study are presented with specific emphasis on their main development features.

Chapter 4 provides an outline of the methodology adopted for this empirical study. It describes the system development tasks involved in the implementation phase as well as the software development strategy used. The implementation phase includes detailed design and code implementation. Other implementation-related activities such as size and cost/effort estimation, data collection, software measurement and statistical analysis methods are also included as an overall base appropriate for the empirical study.

Chapter 5 presents the experimental results of the case study. It describes the detailed analysis of experimental data and discusses related observations associated with the results of the analysis.

Chapter 6 provides a summary and appropriate conclusions based on the empirical study conducted in chapter 5.