INVESTIGATION OF THE RELATIONSHIP BETWEEN VOLUME METRIC AND SYMBOL TABLE SIZE

By

SRIKANTH REDDY MALLADI

Bachelor of Technology

in Electrical and Electronics Engineering

Jawaharlal Nehru Technological University

Hyderabad, India

1993

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of the
requirements for the degree of
MASTER OF SCIENCE
December 1996

INVESTIGATION OF THE RELATIONSHIP BETWEEN

VOLUME METRIC AND SYMBOL TABLE SIZE

Thesis Approved

Mansur Samadzadeh

Thesis Adviser

Blayne E. Mayfield

H. Lu

Thomas C. Collins

Dean of the Graduate College

PREFACE

Software metrics quantify various structural aspects of programs based on abstractions of programs (e.g., control flow graph, token classification, or data dependency graph). Dynamic or execution-dependent measures of programs include number of errors encountered, time/space required for execution, and program symbol table size. There have been several studies investigating possible relationships between structural software metrics and dynamic measurements of programs. This study examines the relationship between the volume of a program as a static metric defined in Software Science and the program's symbol table size as a dynamic measure of its behavior. Various static measures relating to the size of a program were collected using three publicly available tools: HALSTEAD, METRE, and MY_VOL. The symbol table size was measured using two routines: ST_SIZE1 and ST_SIZE2.

The results of this empirical investigation (using a collection of 100 assorted programs) indicate a strong, positive correlation between Halstead's volume metric values and dynamic symbol table sizes among string processing, numeric, and motif programs. This is consistent with the general observation that as program size increases (with the increase in the number of operators and operands), the number of symbols in the symbol table increases also. High correlation coefficients (values greater than 0.5) between volume metric and symbol table size for string processing, numeric, and motif programs provide anecdotal evidence for the general observation above. Because of the differences in the variations of

symbol table size with the variations of volume metric among different classes of programs, the same result was not observed for the general class of programs (i.e., all programs under study). There seems to be little or no correlation between Halstead's volume metric values and the symbol table sizes for the general class of programs. Among other correlations considered in this study, the correlation between operand volume and symbol table size was more prominent than the correlation between operator volume and symbol table size for string processing programs. For motif programs, the correlation between operator volume and symbol table size was more prominent than the correlation between operand volume and symbol table size.

ACKNOWLEDGEMENTS

I would like to thank my advisor and mentor Dr. Mansur H. Samadzadeh who motivated me by his valuable instruction and example. His guidance and constructive comments throughout this thesis work helped me a great deal in adapting myself to the requirements of this undertaking. Dr. Samadzadeh gave generously of his time and expertise for correcting my work and rendering helpful suggestions.

Special thanks also go to my other committee members, Drs. Blayne Mayfield and Huizhu Lu for their cooperation and valuable suggestions.

Also, I would like to express my gratitude to my parents, brothers, sisters, and all my relatives who were mentally with me all the time. My friends deserve equal appreciation for standing behind me, as friends should.

TABLE OF CONTENTS

CHAPTER PAGE

I. INTRODUCTION 1

II. SOFTWARE METRICS 3

2.1 Size Metrics 4
2.1.1 Lines of Code Metric 5
2.1.2 Software Science Metrics 5
2.1.2.1Volume Metric 6
2.2 Dynamic Measures 6

III. SYMBOL TABLE 8

3.1 Symbol Table Implementations 9

IV. EXPERIMENTAL DETAILS 11

4.1 Experiment Definition 12
4.2 Metrics Used in the Experiment 12
4.2.1 Static Measures 12
4.2.2 Dynamic Measures 15
4.3 Resources Used in the Experiment 15
4.3.1Tools to Measure Volume Metric 16
4.3.1.1 HALSTEAD 16
4.3.1.2 METRE V 2.3 17
4.3.1.3 MY_VOL 17
4.3.2 Methods Used to Measure Symbol Table Size 17
4.3.2.1 ST_SIZE1 18
4.3.2.2 ST_SIZE2 18
4.3.3 CSIZE 19
4.4 Test Suite 20
4.5 Data Collection 21

V. ANALYSIS OF MEASUREMENTS 23

5.1 String Processing Programs 25
5.2 Network Programs 31
5.3 Numeric Programs 35

CHAPTER PAGE

5.4 Graphics Programs 40
5.5 Motif Programs 43

VI. SUMMARY, CONCLUSIONS, AND FUTURE WORK 50

REFERENCES 53

APPENDICES 56

APPENDIX A - GLOSSARY AND TRADEMARK INFORMATION 57
APPENDIX B - SOURCE AND BRIEF DESCRIPTION OF PROGRAMS IN THE TEST SUITE 60
APPENDIX C - OUTPUT OF CORRELATION ANALYSIS FOR ALL PROGRAMS IN THE TEST SUITE 66
APPENDIX D - DYNAMIC AND STATIC MEASURES FOR INDIVIDUAL APPLICATION AREAS 81
APPENDIX E - LEX SPECIFICATION AND SHELL SCRIPT USED IN THE HALSTEAD TOOL 110
APPENDIX F - LEX SPECIFICATION AND SHELL SCRIPT USED IN THE MY_VOL TOOL 115
APPENDIX G - ST_SIZE1- A ROUTINE TO COMPUTE SYMBOL TABLE SIZE 120
APPENDIX H - ST_SIZE2 - A ROUTINE TO COMPUTE SYMBOL TABLE SIZE 122

LIST OF TABLES

TABLE PAGE

1. A typical symbol table representation 8

2. Number of programs obtained from each source 21

3. Classification of programs and the number of programs of each type 21

4. Correlations between symbol table sizes as obtained by routines st-size1 and st-size2 24

5. Correlations between modularV values obtained by using the three tools HALSTEAD, METRE, and MY_VOL 24

6. Statistical measures for static and dynamic metrics for the 29 string processing programs in the test suite 26

7. Correlations between dynamic measures and static measures obtained by using HALSTEAD and CSIZE for the string processing programs 28

8. Correlations between dynamic measures and static measures obtained by using METRE and CSIZE for the string processing programs 31

9. Correlations between dynamic measures and static measures obtained by using MY_VOL and CSIZE for the string processing programs 31

10. Statistical measures for static and dynamic metrics for the 13 network programs in the test suite 33

11. Correlations between dynamic measures and static measures obtained by using HALSTEAD and CSIZE for the network programs 34

12. Correlations between dynamic measures and static measures obtained by using METRE and CSIZE for the network programs 34

13. Correlations between dynamic measures and static measures obtained by using MY_VOL and CSIZE for the network programs 34

CHAPTER PAGE

14. Statistical measures for static and dynamic measures for the 17 numeric programs in the test suite 38

15. Correlations between dynamic measures and static measures obtained by using HALSTEAD and CSIZE for the numeric programs 39

16. Correlations between dynamic measures and static measures obtained by using METRE and CSIZE for the numeric programs 40

17. Correlations between dynamic measures and static measures obtained by using MY_VOL and CSIZE for the numeric programs 40

18. Statistical measures for static and dynamic metrics for the 61 graphics programs in the test suite 41

19. Correlations between dynamic measures and static measures obtained by using HALSTEAD and CSIZE for the graphics programs 42

20. Correlations between dynamic measures and static measures obtained by using METRE and CSIZE for the graphics programs 42

21. Correlations between dynamic measures and static measures obtained by using MY_VOL and CSIZE for the graphics programs 43

22. Statistical measures for static and dynamic metrics for the 46 Motif programs in the test suite 45

23. Correlations between dynamic measures and static measures obtained by using HALSTEAD and CSIZE for the Motif programs 46

24. Correlations between dynamic measures and static measures obtained by using METRE and CSIZE for the Motif programs 48

25. Correlations between dynamic measures and static measures obtained by using MY_VOL and CSIZE for the Motif programs 48

LIST OF FIGURES

FIGURE PAGE

1. A COFF file's basic structure 19

2. File header (struct filehdr) structure declaration in filehdr.h 19

3. Plot of $\log_{10}$(st-size1) vs. $\log_{10}$(V) for the string processing programs 28

4. Plot of $\log_{10}$(st-size1) vs. $\log_{10}$(V) for the numeric programs 39

5. Plot of $\log_{10}$(st-size1) vs. $\log_{10}$(V) for the Motif programs 46

CHAPTER I

INTRODUCTION

The bulk of software documents and the complexity of processes that make up modern-day software projects demand a structured design and planning much before the coding phase. Applying software engineering principles and practices has become a necessity for efficient testing, development, and maintenance of software projects.

Software metrics as a sub-field of software engineering is concerned with defining and validating measurable values that correspond to the abstract structural characteristics of programs. Measurements of various aspects of software give a quantitative picture of the quality and quantity of the software products delivered. Size of a program is an important software metric for quantitative analysis of computer programs. There are a number of ways of defining the size of a program. The number of lines of executable code and volume [Halstead 77a] are two such metrics. The volume metric, chosen to characterize the size property of a program, is independent of the character set of a program [Halstead 77a].

Dynamic measures, which generally refer to the compile time or execution time measures, are indicators of the process of program development or the behavior of programs after the coding phase is completed. They include the number of changes made or errors

encountered, the time/space required to execute programs, and program symbol table sizes. As a compiler translates a program in a high-level programming language into a low-level language, various dynamic measures can be obtained as by-products, e.g., the number of errors encountered or the symbol table size.

Though there have been some studies correlating static and dynamic measures [Moll and Samadzadeh 89] [Shaw et al. 89] [Koskimies and Paakki 87] [Plattner and Nievergelt 81], the relationship between the volume metric and the symbol table size had not been addressed in the open literature. This study aims to build a hypothesis correlating the static volume measure of a program with its symbol table size as one dynamic measure of program behavior. The major areas identified for this study include Halstead's software science metrics for the volume metric and symbol table implementation for dynamic measurements. This thesis report discusses each of these areas. Chapter 2 discusses software metrics and the volume metric in particular. Chapter 3 describes the purpose and implementation of symbol tables. Chapter 4 gives the experimentation details. Chapter 5 explains the analysis of the measurements. Summary and future work are discussed in Chapter 6.

CHAPTER II

SOFTWARE METRICS

Software metrics may be defined as indicators of software products or the software development process that provide feedback to developers and maintainers, giving them some control over projects. They are used for measuring abstract structural characteristics of software in general.

Metrics most often used in software evaluation are lines of code, Halstead's software science metrics [Halstead 77a], McCabe's complexity [McCabe 76], and Albrecht's function point metric [Albrecht 79]. Other metrics include the knot count metric [Woodward et al. 79], amount of data [Conte et al. 86], variable spans [Dunsmore and Gannon 79], nesting levels [Zweben and Fung 79], interconnectivity [Kafura and Henry 81], and the residual complexity metric [Samadzadeh and Edwards 88].

Software metrics can be classified in different ways. A classification offered by Perlis et al. [Perlis et al. 81] categorizes metrics as human performance metrics and program performance metrics. Human performance metrics deal with the development phase of the software, whereas program performance metrics deal with the application phase of the software. Conte et al. [Conte et al. 86] categorized metrics as either process or product

metrics, depending on whether they measure the development process and environment or the software product, respectively. A classification offered by Basili and Hutchens [Basili and Hutchens 83] is identified to be most suitable for this study. They divided software metrics into static and dynamic classes. Static measures are concerned with measurements relating to software during the coding phase. They are collected by performing static analysis on the source code. Dynamic measures are concerned with measurements relating to software products released after the coding phase. They are collected at run time or compile time. Subsection 2.1 below deals with the size metrics including the volume metric (one of Halstead's software science metrics), which fall into the static metrics class. Various dynamic measures are presented in Subsection 2.2.

2.1 Size Metrics

One of the important concerns of software metrics is measurement of various aspects of the static size of a software product. Size metrics can be regarded as a subcategory of static metrics, which are generally based on the surface structural properties of a software product such as data organization, size, and control organization. Specifically, size metrics deal with a measure of the size of a software product (e.g., number of lines of code, number of functions, or number of tokens). Halstead [Halstead 77a] proposed a comprehensive set of software metrics that can be mostly classified as size metrics.

There are many models described in the literature to measure software size or bulk [Wolverton 74] [McCabe 76] [Halstead 77a] [Albrecht 79]. However, a perfect estimation for size is difficult to derive. Any derived metric will be less reliable when the environment

for which it is designed changes [Verner and Tate 92]. Conte et al. summarized this uncertainty in the size metric as follows [Conte et al. 86].

> Expert sizing depends on so many subjective factors that different 'experts' can arrive at radically different estimates. This underscores the need for more objectivity based sizing techniques.

The lines of code metric and Halstead's volume metric were selected to be used as size metrics for this study. The lines of code metric is still one of the most widely used metrics in the software engineering community. Software science metrics are commonly used for evaluating software complexity in terms of size. Although the theoretical validations of these metrics have been somewhat disputed in the literature, they are chosen for this study because of the large amount of independently collected empirical data supporting them.

2.1.1 Lines of Code Metric

Although lines of code metric is used frequently in the software engineering community, there is no universally accepted definition for it. Lines of code may be defined in simple terms as the physical number of lines of source code. Lines of code is not considered to be an attribute of software quality [Ejiogu 91], because it generally favors bulkiness over quality. However, since no other metric has gained comparable acceptance as an indicator of size, the lines of code metric continues to be used as a base measure of software size and as a comparative basis for validating other software metrics.

2.1.2 Software Science Metrics

Software science metrics [Halstead 77a] are used to measure various aspects of a program including its bulk in terms of basic token counts. They are based on the contention that "an algorithm consists of operators and operands, and nothing else" [Halstead 77a]. The

rest of this subsection contains some of the software science definitions from the three main references [Conte et al. 86] [Halstead 77a] [Moll and Samadzadeh 89].

The four basic software science metrics are listed below.

n_1 number of unique operators
n_2 number of unique operands
N_1 total number of operators
N_2 total number of operands

Vocabulary size, n, of a program is defined as follows.

$$n = n_1 + n_2$$

Program length, N, is the total number of operators plus the total number of operands.

$$N = N_1 + N_2$$

Program volume, V, is defined as a function of length and vocabulary size.

$$V = N \log_2 n$$

2.1.2.1 Volume Metric The volume of a program, as defined above, can be interpreted as "the fewest number of binary digits or bits with which it could be represented" [Halstead 77a] in a uniform encoding of the program tokens. As such, the estimated volume of a program using Halstead's volume metric calculation is essentially independent of the number of characters (i.e., token identifiers including variable names) in the program. Program volume, V, can be defined in terms of the four basic Halstead metric counts as follows.

$$V = (N_1 + N_2) \log_2(n_1 + n_2)$$

2.2 Dynamic Measures

Dynamic measures are measurements of the behavior of programs at compile

time/run-time. They include the number of errors encountered, execution time, memory used during execution, and number of statements executed [Moll and Samadzadeh 89]. Symbol table size during execution can also be included with the above. A brief description of these measures is given below.

1. The number of errors encountered is the number of syntactic errors and semantic errors detected/corrected through compilation and re-execution.

2. Execution time is the amount of CPU time a program takes to process its input and run to completion, either normally or abnormally.

3. Memory used during execution is the amount of memory a program occupies while executing. This memory space includes the program segment, the data segment, and the stack segment.

4. The number of statements executed is the number of source statements executed, which obviously depends on each specific input to a program.

5. Symbol table size during execution is the size of the symbol table of a program while executing. Measures include various counts associated with the symbol table. Chapter 3 deals with symbol tables in some detail.

This study is concerned with measurements of symbol table size at compile time.

CHAPTER III

SYMBOL TABLE

A symbol table can be viewed as a data structure associating a set of identifiers with their attributes [Pittman and Peters 92]. It stores the information relating to tokens, which is used by the syntax analyzer as well as the code generator. This information in the symbol table is stored and accessed continually at compile time. A typical symbol table representation is given in TABLE 1 [Tremblay and Sorenson 85]. It consists of the name, dimension, and other attributes of tokens generated by a lexical analyzer.

TABLE 1. A typical symbol table representation

Variable	Address	Type	Dimension	Line Declared	Other Attributes
Company	0	2	1	2	//////
X3	4	1	0	3	//////
Form1	8	3	2	4	//////
B	48	1	0	5	//////
Ans	52	1	0	5	//////
M	56	6	0	6	//////
First	64	1	0	7	//////

Operations performed on a symbol table include inserting and finding token information. When a variable is declared, its attributes are entered in the symbol table. These

attributes are recalled when that variable is referenced in the source program [Aho et al. 86]. Various data structure organizations are implemented for efficient performance of the operations performed on a symbol table. Accordingly, symbol tables are classified into a number of different types. The simplest classification [Tremblay and Sorenson 85] is to divide symbol tables into unordered symbol tables and ordered symbol tables. In unordered symbol tables, token information is inserted in the order in which the tokens are first encountered during compilation. On the other hand, in ordered symbol tables, token information is inserted in the lexicographical order of variable names.

3.1 Symbol Table Implementations

Symbol tables can be implemented with different search strategies depending on various attributes of the tokens. When the number of variables appearing in a program is small, an unordered symbol table implementation with linear search is preferred. If the number of variables in a program is large, an ordered symbol table implementation with binary search strategy can be employed [Mak 91]. If the frequency of the insertion operation for variables is large, a tree-structured symbol table performs better because of the relative ease of handling insertions. If the availability of memory is not a problem, hash symbol table implementation is considered to be the best implementation [Tremblay and Sorenson 85].

Two types of storage-allocation strategies are implemented with symbol tables, static and dynamic [Aho et al. 86]. In static storage allocation, variable-length strings for identifiers, and nested procedures and recursion are not allowed in the source code. Once compiled, the symbol table size remains constant and does not vary during execution. The

dynamic storage-allocation strategy does not impose any limit on the length of strings for identifiers, and nested procedures and recursion are allowed in the source code. In dynamic storage allocation, as the execution sequence enters a nested loop or as a recursive call occurs, a special data area is created over the existing stack of the symbol table. As such, the symbol table size varies with the number of nested loops and recursive calls during program execution.

CHAPTER IV

EXPERIMENTAL DETAILS

This section describes the experiment conducted in investigating the relationship between program volume and symbol table size. The framework employed consisted of defining the purpose of the experiment, planning according to available resources, acquiring/developing tools that aid in the implementation, conducting the experiment, collecting data, and analyzing the results. The major purpose of this exploratory and prototypical experiment can be stated in Perlis' words: "although an experiment does not prove a hypothesis, it does allow for the rejection of competing alternative explanations of a phenomenon" [Perlis et al. 81]. As such, an experiment's results, say, may not necessarily show that there is a positive (or negative) correlation between the volume and the symbol table size, but they may aid in considering alternatives for measuring the program volume. The following subsections describe the definition of the experiment, definitions of the metrics used, tools and methods used to measure the volume metric and the symbol table size, the test suite used for the experiment, and the data collection phase of the experiment.

4.1 Experiment Definition

The main thrust of the proposed study was to investigate the possibility of a correlation between the static volume of a program and its symbol table size during execution. The work done consisted of conducting an exploratory empirical study of a number of programs from the perspective of volume and symbol table size, and finding a possible correlation between the two measures.

4.2 Metrics Used in the Experiment

This subsection explains the static and dynamic metrics used in the experiment.

4.2.1 Static Measures

These measures were collected from the static source code of programs in the test suite. In this experiment, a program is considered as a collection of modules that can be compiled to produce an object file. A module is considered as an individual entity of a program which can be compiled independently of the other entities. It may be any file with a '.c' or a '.h' extension.

The static measures for a program are based on the following nine measures for each module.

1.a n_{1m} is the number of unique operators in a module.

2.a n_{2m} is the number of unique operands in a module.

3.a n_m is the number of unique tokens in a module, which is the sum of n_{1m} and n_{2m}.

4.a N_{1m} is the total number of operators in a module.

5.a N_{2m} is the total number of operands in a module.

6.a N_m is the total number of tokens in a module, which is the sum of N_{1m} and N_{2m}.

7.a V_m is volume of a module as defined below.

$$V_m = N_m \log_2(n_m)$$
or
$$V_m = (N_{1m} + N_{2m}) \log_2(n_{1m} + n_{2m})$$

8.a optrV_m is operator volume of a module as defined below.

$$\text{optr}V_m = N_{1m} \log_2(n_{1m})$$

9.a opndV_m is operand volume of a module as defined below.

$$\text{opnd}V_m = N_{2m} \log_2(n_{2m})$$

Based on the module level measures defined above, the static measures for a program with j modules can be defined as follows.

1.b Number of unique operators in a program, modular_n_1, is as defined below.

$$\text{modular\_}n_1 = \sum_{i=1}^{j} n_{1m}^{i}$$

2.b Number of unique operands in a program, modular_n_2, is as defined below.

$$\text{modular\_}n_2 = \sum_{i=1}^{j} n_{2m}^{i}$$

3.b Number of unique tokens in a program, modular_n, is as defined below.

$$\text{modular\_}n = \sum_{i=1}^{j} n_{m}^{i}$$

4.b Total number of operators in a program, modularN_1, is as defined below.

$$\text{modular}N_1 = \sum_{i=1}^{j} N_{1m}^{i}$$

5.b Total number of operands in a program, modularN_2, is as defined below.

$$\text{modular}N_2 = \sum_{i=1}^{j} N_{2m}^{i}$$

6.b Total number of tokens in a program, modularN, is as defined below.

$$\text{modularN} = \sum_{i=1}^{j} N_m^i$$

7.b Sum of volume of modules, modularV, is as defined below.

$$\text{modularV} = \sum_{i=1}^{j} V_m^i$$

8.b Sum of operator volume of modules, modular_optrV, is as defined below.

$$\text{modular\_optrV} = \sum_{i=1}^{j} \text{optrV}_m^i$$

9.b Sum of operand volume of modules, modular_opndV, is as defined below.

$$\text{modular\_opndV} = \sum_{i=1}^{j} \text{opndV}_m^i$$

10. Sum of operator volume of modules and operand volume of modules, modular_sumV, is as defined below.

$$\text{modular\_sumV} = \text{modular\_optrV} + \text{modular\_opndV}$$

11. Volume of a program, V, is as defined below.

$$V = (\text{modularN}_1 + \text{modularN}_2) \log_2(\text{modular\_n}_1 + \text{modular\_n}_2)$$

12. Operator volume of a program, optrV, is as defined below.

$$\text{optrV} = \text{modularN}_1 \log_2(\text{modular\_n}_1)$$

13. Operand volume of a program, opndV, is as defined below.

$$\text{opndV} = \text{modular\_N}_2 \log_2(\text{modular\_n}_2)$$

14. Sum of operator volume and operand volume of a program, sumV, is as defined below.

$$\text{sumV} = \text{optrV} + \text{opndV}$$

15. Lines Of Code measure, LOC, is the sum of the number of non-blank, non-commentary lines in all the modules. This measure is obtained by using the CSIZE tool (described in Subsection 4.3.3).

4.2.2 Dynamic Measures

Symbol table size is the only dynamic measure used in this experiment. This measure is obtained after the compilation of the source code using 'cc' or 'gcc' compilers. It is the size of the symbol table in terms of the number of symbol table entries as obtained by using the two methods described in Subsection 4.3.2.

1. st-size1 indicates the symbol table size obtained by using the ST_SIZE1 method.
2. st-size2 indicates the symbol table size obtained by using the ST_SIZE2 method.

4.3 Resources Used in the Experiment

The initial part of the experiment consisted of selecting appropriate tools to measure volume metric and symbol table size. Various tools widely used in the software engineering community to measure static and dynamic measures were studied and examined from the point of view of their suitability to the experiment and their portability to the Sequent Symmetry S/81 system, which was the platform for the experiment. Various DYNIX/ptx utilities available on the Sequent Symmetry S/81 system in the Computer Science Department at OSU, including various shell commands [Bourne 78] and profiling tools such as `prof`, `nm`, and `gprof` were investigated to aid in obtaining quantitative information from symbol tables. Various tools described in the literature for obtaining program profiles were also investigated, including tools to collect run-time statistics based on execution profile monitoring techniques presented by Bishop [Bishop 87] and Graham et al. [Graham et al. 83]. The tools and methods selected are described in the following subsections.

4.3.1 Tools to Measure Volume Metric

The volume metric is an important software science metric. This measure is widely used in the software engineering community for collecting information relating to the size of software. Various tools (from vendors or on a shareware basis) are available for collecting the volume metric. Two tools, HALSTEAD and METRE (both available at the archive sites of `comp.software-eng`), were used for this experiment.

Another tool, MY_VOL, was developed based on the HALSTEAD tool by making a few changes in the parsing rules. There were two main differences between the selected tools in the strategies employed for computing the volume metric. The first difference was in the operator and operand classification. There is no general agreement in the software engineering community on this classification [Shen et al. 83]. The second difference was in the way the volume metric was computed for a program with more than one procedure in each module. A few modifications were made to the software tools to include some more static measures (as defined in Subsection 4.2.1). The following subsections describe each of these three tools in some detail.

4.3.1.1 HALSTEAD This tool comes as part of the "Cmetrics" package available at the archives of the usenet newsgroup `comp.software-eng`. HALSTEAD measures various software metrics of an input program written in the C programming language. The outline of the algorithm for this tool is as follows.

1. Strip comments from the given input C program.
2. Remove Strings.
3. Remove delimiters.

4. Break input into tokens.

5. Count total number of tokens as token count.

6. Count unique tokens as vocabulary size.

7. Calculate volume = (token count) $\log_2$(vocabulary size).

The lex specification for this tool and the modified shell script is given in Appendix E. This tool is simple to use and is portable to the Sequent Symmetry S/81 machine.

4.3.1.2 METRE V 2.3 This tool is copyrighted to Paul Long, and is available at the archives of the usenet newsgroup `comp.software-eng`. It uses a standard C (ANSI/ISO) parser whose behavior is determined by a set of rules. It measures various software metrics such as Function Points, McCabe's Cyclomatic Complexity, LOC, and Halstead's metrics (including the volume metric) for preprocessed C programs. A korn shell program was written to preprocess the C modules, and then the METRE tool was executed on each preprocessed module. A few modifications were made to this tool to compute various other measures (as described in Subsection 4.2.1) useful in the study.

4.3.1.3 MY_VOL This tool was developed by the author based on the HALSTEAD tool. The main modification being the change in the lex specification. It uses a standard C (ANSI/ISO) parser. The lex specification and the shell script for this tool are given in Appendix F.

4.3.2 Methods Used to Measure Symbol Table Size

The symbol table size was measured as the number of symbol table entries. Two methods to measure the symbol table size were developed. They are based on the utilities available on the Sequent Symmetry S/81 machine in the Computer Science Department at

OSU. These two methods are described in the following subsections.

4.3.2.1 ST_SIZE1 This method uses the information about the symbol table stored in the file header structure (struct filehdr) of any machine code executable file that follows the COFF (Common Object File Format) structure definition. A COFF file's basic structure is given in Figure 1 [O'Reilly 88]. The file header structure is given in Figure 2.

A routine was developed which extracts the number of symbol table entries (f_nsyms) from the file header section of the input file which follows the COFF structure definition. This routine is listed in Appendix G.

4.3.2.2 ST_SIZE2 This method was developed using the 'nm' utility available on the Sequent Symmetry S/81 machine. The 'nm' utility dumps the symbol table of each common object file given as input. Then number of symbol table entries was calculated using the following two commands.

```
nm -hfT <file> <temp>
wc -l <temp>
```

The first command dumps the symbol table of <file> in <temp>. The '-h' option suppresses the output header. The '-f' option produces full output without suppressing any symbols that are normally suppressed. The '-T' option truncates the names of symbols so that only one symbol per line is printed. The second command counts the number of lines in the <temp> file which is the number of symbol table entries.

A korn shell routine was written to compute the symbol table size for group of files given as input on command line. This routine is listed in Appendix H.

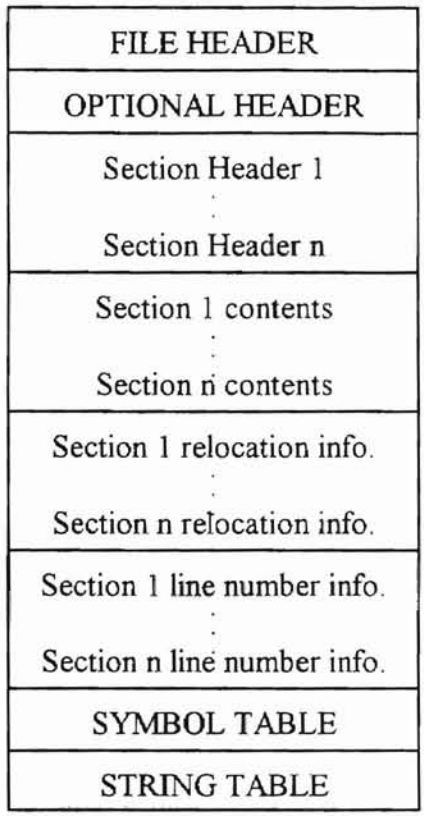

Figure 1. A COFF file's basic structure

```
struct filehdr
{
  unsigned short  f_magic;  /* magic number */
  unsigned short  f_nscns;  /* number of sections */
  long            f_timdat; /* time & date stamp */
  long            f_symptr; /* file pointer to symtab */
  long            f_nsyms;  /* number of symtab entries */
  unsigned short  f_opthdr; /* size of (optional hdr) */
  unsigned short  f_flags;  /* flags */
};
```

Figure 2. File header (struct filehdr) structure declaration in filehdr.h

4.3.3 CSIZE

This tool was developed by Christopher Lott to measure the size of C source code.

It is available at the archives of the usenet newsgroup `comp.software-eng` (`ftp.qucis.queensu.ca`). It gives as output the total number of lines, blank lines, lines with comments, non-blank and non-comment lines, semicolons, and preprocessor directives of C source code. The Lines Of Code metric, LOC, was computed as the number of non-blank, non-comment lines as given by this tool.

4.4 Test Suite

The test suite for this experiment mainly consisted of the programs available on the Sequent Symmetry S/81 machine in the Computer Science Department at OSU. The `'/contrib'` and `'/usr/local'` directories contain a collection of standard and non-standard programs, utilities, and games contributed to the system by its users. These programs vary in size from a few hundred lines of code to thousands of lines of code, and represent various fields of Computer Science. Programs taken from the public domain of various ftp sites were also used in the test suite. Motif programs from `ftp.uu.net` were also included in the test suite. Other programs in the test suite included a few programs written by the author in the Computer Networks class (COMSC 4283). The number of programs obtained from each source is given in TABLE 2.

Since the programs in the test suite were taken from various sources, the test suite can be assumed to be a first approximation to a reasonably representative sample of correct and production quality application programs. A few minor modifications were made to these programs to be able to compile them with the 'cc' and 'gcc' compilers available on the Sequent Symmetry S/81 machine. The modifications were restricted to the makefiles that

came along with the source codes of the programs. No modifications were made to the source codes of the programs. TABLE 3 gives a classification of the programs based on various programming fields and the number of programs of each type. The source of the programs in the test suite and a brief description of each program is given in Appendix B.

TABLE 2. Number of programs obtained from each source

Source	Number of Programs
'/contrib/src'on Sequent	25
'/local/src' on Sequent	13
'/usr/sequent/tcp_examples' on Sequent	11
comp.language.c usenet newsgroup archive	2
programs written by the author in the Computer Networks class (COMSC 4283)	3
motif programs obtained from ftp.uu.net	46

TABLE 3. Classification of programs and the number of programs of each type

Program Type	Number of Programs
string processing	29
numeric	17
network	13
graphics	61
motif	46

4.5 Data Collection

This section describes the data collection phase of the experiment. Data for analysis was collected on the Sequent Symmetry S/81 machine in the Computer Science Department at OSU using the tools described in Subsection 4.3. The data collection phase was divided into two subphases, data collection for static measures and data collection for dynamic

measures as decribed in the following two paragraphs.

The static measures, defined in Subsection 4.2.1, were obtained using the three tools described in Subsection 4.3.1. Actually, a korn shell script was written to apply the three tools to each program in the test suite and produce the output in a format suitable for analysis using the Microsoft Excel 5.0 statistical package.

The programs in the test suite were individually compiled on the Sequent Symmetry S/81 machine. As these programs had been collected from different sources, each program had to be compiled individually by making the necessary changes to the corresponding makefiles. The changes made in the makefiles include deleting the '-s' flag for the loader, which removes the symbol table structure from the object file, adding the '-O' flag for the compiler for optimization, and other changes pertaining particularly to the DYNIX/ptx operating system on the Sequent Symmetry S/81 machine. All the compilations were made using 'cc', the ANSI C compiler, or 'gcc', the GNU C compiler. These compilations produced an executable object file from which the symbol table size in terms of the number of symbol table entries was extracted by using the methods described in Subsection 4.3.2.

CHAPTER V

ANALYSIS OF MEASUREMENTS

The data collected on the Sequent Symmetry S/81 machine was analyzed using the Microsoft Excel 5.0 statistical package. Detailed correlation analysis was carried out for each pair of measures for possible relationships between them. Significance levels were obtained by referring to the table for critical values for the product-moment correlation coefficient [Conte et. al 86].

The two primary sets of data examined were dynamic symbol table size and static volume metric. As stated in earlier chapters, two methods were used to measure the symbol table size (Subsection 4.3.2) and four tools were used to measure various static measures relating to the volume metric (Subsection 4.3.1). As the first step in the analysis, scatter diagrams were used to approximate the relationship between the two sets of measures. Those diagrams suggested a generally nearly linear relationship between the two sets.

The data in the two sets were examined to see if they satisfied the parametric tests [Conte et al. 86]. The observations were independent, had nearly the same standard deviation, and were at least on the interval scale. They were assumed to have been drawn from a normally distributed population as the values of skewness and kurtosis for the

observations were almost the same as those for the normal distribution.

Pearson product-moment correlation analysis was used to see if the values obtained by using the two methods to compute the symbol table size correlated well with each other. As expected, a positive correlation coefficient of 0.9981 was obtained between the two measures. This value is significant at the 0.01 level of significance for 100 observations. This indicates that one measure varies at the same rate as the other. The correlation coefficients are listed in TABLE 4.

TABLE 4. Correlations between symbol table sizes as obtained by routines st-size1 and st-size2

	st-size1	*st-size2*
st-size1	1.0000	
st-size2	0.9981	1.0000

Similarly, Pearson product-moment correlation analysis was used to check for variation in the values obtained by using the three tools to compute static metrics (as defined in Subsection 4.2.1). Though the measures from each tool were different, the correlation analysis yielded a strong, positive correlation coefficient between each pair. These values are significant at 0.01 level of significance. This shows that there is little variation in the values obtained by the three tools. The correlation coefficients between the values obtained for modularV by the three tools are listed in TABLE 5.

TABLE 5. Correlations between modularV values obtained by using the three tools HALSTEAD, METRE, and MY_VOL

	HALSTEAD	*METRE*	*MY_VOL*
HALSTEAD	1.0000		
METRE	0.9819	1.0000	
MY_VOL	0.9997	0.9816	1.0000

The output of the correlation analysis made for the static and dynamic measures collected for all the programs in the test suite is listed in Appendix C. The Pearson product-moment correlation coefficients obtained were not significant at 0.05 level of significance for any pair of measurements of concern. These results prompted for a classification of programs based on programming field or application area, i.e., string processing, network, numeric, graphics, and Motif programs. The following subsections explore the results obtained from correlation analysis for each class (or application area) of the programs in the test suite.

5.1 String Processing Programs

The two sets of data, i.e., static and dynamic measures, obtained for this class of programs were found to be parametric. Various statistical measures for the static and dynamic observations, which were obtained by using the six different software tools (see Subsection 4.3) for the string processing programs, are listed in TABLE 6.

The scatter diagram suggested a linear relationship between the two sets of data. As such, Pearson product-moment correlation analysis was used to assess the strength of the linear relationship between the two sets of measures. A positive correlation coefficient of more than 0.5 was observed between st-size1 and all the static measures obtained by using the HALSTEAD tool. All the observations made were found to be significant at the 0.01 level of significance. The highest correlation coefficient of 0.5951 was observed between modular_n_2 and st-size1. The correlation coefficient between program volume, V, and st-size1 was 0.5249, indicating that there is a positive correlation between program volume obtained by using the HALSTEAD tool and the symbol table size obtained by using

ST_SIZE1 for the class of programs that fall under string processing programs. The plot of $\log_{10}$(st-size1) vs. $\log_{10}$(V) is shown in Figure 3. High correlation coefficients of 0.5111, 0.5065, and 0.5238 were also observed for modularV, modular_sumV, and sumV with st-size1, respectively. Similarly, positive correlations were observed between st-size2 and all the static measures obtained by using the HALSTEAD tool. The correlation coefficients are listed in TABLE 7.

TABLE 6. Statistical measures for static and dynamic metrics for the 29 string processing programs in the test suite

Statistical measures for static measures obtained by using HALSTEAD

	modular_n_1	modular_n_2	modularN$_1$	modularN$_2$	modularV	modular_sumV	modular_optrV	modular_opndV	V	sumV	optrV	opndV
Minimum	32	40	145	91	1456	1209	725	484	1456	1209	725	484
Maximum	1696	5661	31440	23301	427183	337901	173270	168418	674036	603466	312976	290490
Mean	315.55	1184.86	8762.07	6291.21	119737.21	95415.10	47206.79	48207.83	170033.69	147543.41	78426.86	69116.62
Median	153	675	5051	3401	66209	54168	27725	26076	83950	69904	37588	32316
Standard Deviation	365.67	1368.83	9615.46	7093.27	135246.44	107544.71	52252.61	55516.53	205325.98	180469.25	95616.63	85114.57
Kurtosis	6.25	3.25	0.28	0.27	0.13	0.10	0.25	0.11	0.75	0.86	0.92	0.89
Skewness	2.21	1.82	1.19	1.19	1.17	1.16	1.18	1.17	1.36	1.38	1.40	1.39

Statistical measures for static measures obtained by using METRE

	modular_n_1	modular_n_2	modularN$_1$	modularN$_2$	modularV	modular_sumV	modular_optrV	modular_opndV	V	sumV	optrV	opndV
Minimum	22	32	81	61	817	666	361	305	817	666	361	305
Maximum	4100	6633	27953	17311	271347	216146	123720	92411	606074	555246	335475	219771
Mean	765.52	1296.59	6813.83	4593.66	72079.59	56922.17	30487.21	26436.14	134680.21	121436.59	70256.55	51180.10
Median	253	456	3063	2180	32675	26192	14035	12160	49901	44108	23688	19256
Standard Deviation	1067.49	1711.99	7909.35	5215.36	80463.80	63234.77	34303.85	29110.55	171136.88	156569.26	92473.70	64271.71
Kurtosis	3.02	3.23	0.94	0.34	0.18	0.13	0.59	-0.19	1.43	1.51	1.85	1.07
Skewness	1.87	1.91	1.33	1.18	1.13	1.11	1.20	1.04	1.48	1.50	1.58	1.40

Statistical measures for dynamic measures and static measures obtained by using MY_VOL and CSIZE

	modular_n	modularN	modularV	V	LOC	st-size1	st-size2
Minimum	73	237	1467	1002	51	781	441
Maximum	7355	53759	418341	690508	11769	10305	5634
Mean	1502.10	15117.31	120138.69	170590.28	2846.66	3272.48	1828.10
Median	807	8290	66103	81006	1430	2285	1260
Standard Deviation	1720.74	16792.37	135894.91	206959.59	3178.10	2586.35	1414.45
Kurtosis	3.83	0.21	0.10	0.77	1.27	1.03	1.07
Skewness	1.89	1.17	1.16	1.35	1.41	1.22	1.23

Other observations made from the correlation analysis of string processing programs for static measures, obtained by using HALSTEAD, and dynamic measures were:

1. Correlations between the number of unique operands, modular_n_2, vs. st-size1 and st-size2 (correlation coefficients of 0.5951 and 0.6000, respectively) were more prominent than the correlations between the number of unique operators, modular_n_1, vs. st-size1 and st-size2 (correlation coefficients of 0.5119 and 0.5189, respectively).

2. Correlations between the total number of operands, modularN_2, vs. st-size1 and st-size2 (correlation coefficients of 0.5279 and 0.5341, respectively) were more prominent than the correlations between the total number of operators, modularN_1, vs. st-size1 and st-size2 (correlation coefficients of 0.5002 and 0.5064, respectively).

3. Correlations between the sum of operand volumes of modules, modular_opndV, vs. st-size1 and st-size2 (correlation coefficients of 0.5305 and 0.5362) were more prominent than the correlations between the sum of operator volumes of modules, modular_optrV, vs. st-size1 and st-size2 (correlation coefficients of 0.4788 and 0.4857, respectively).

4. Correlations between the operand volume of a program, opndV, vs. st-size1 and st-size2 (correlation coefficients of 0.5427 and 0.5485, respectively) were more prominent than the correlations between the operator volume of a program, optrV, vs. st-size1 and st-size2 (correlation coefficients of 0.5055 and 0.5121, respectively).

5. Program volume, V, correlated more with st-size2 (correlation coefficient of 0.5309) than with st-size1 (correlation coefficient of 0.5249).

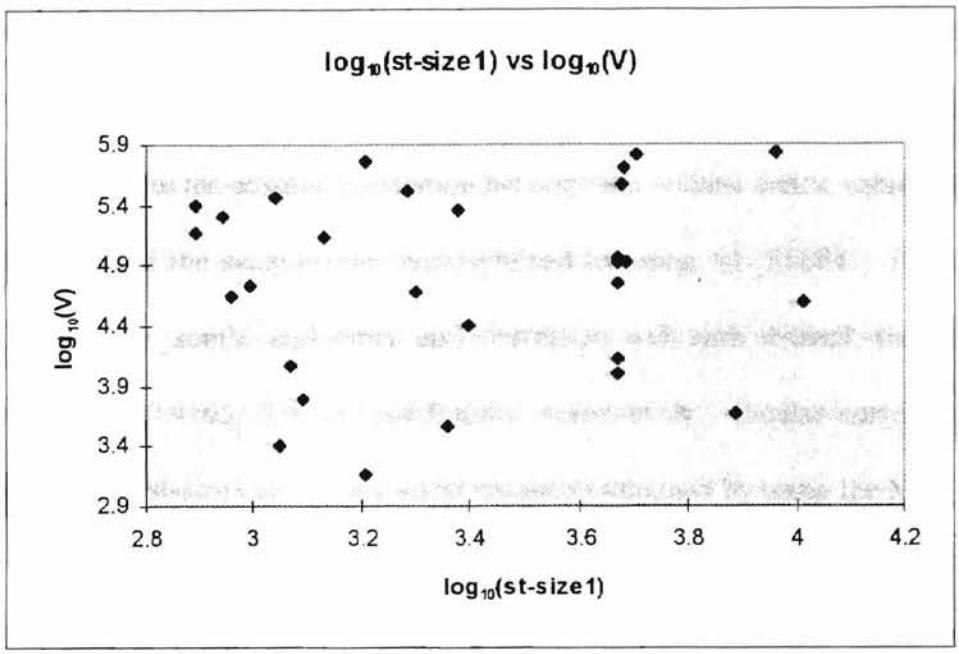

Figure 3. Plot of $\log_{10}$(st-size1) vs. $\log_{10}$(V) for the string processing programs

TABLE 7. Correlations between dynamic measures and static measures obtained by using HALSTEAD and CSIZE for the string processing programs

	st-size1	st-size2	modular_n_1	modular_n_2	modularN$_1$	modularN$_2$	modularV	modular_sumV	modular_optrV	modular_opndV	V	sumV	optrV	opndV	LOC
st-size1	1.0000														
st-size2	0.9984	1.0000													
modular_n_1	0.5119	0.5189	1.0000												
modular_n_2	0.5951	0.6000	0.9651	1.0000											
modularN$_1$	0.5002	0.5064	0.8701	0.9415	1.0000										
modularN$_2$	0.5279	0.5341	0.8947	0.9561	0.9936	1.0000									
modularV	0.5111	0.5170	0.8491	0.9322	0.9973	0.9932	1.0000								
modular_sumV	0.5065	0.5128	0.8534	0.9342	0.9977	0.9943	0.9998	1.0000							
modular_optrV	0.4788	0.4857	0.8447	0.9256	0.9983	0.9881	0.9975	0.9978	1.0000						
modular_opndV	0.5305	0.5362	0.8582	0.9385	0.9931	0.9960	0.9980	0.9980	0.9917	1.0000					
V	0.5249	0.5309	0.9014	0.9646	0.9953	0.9965	0.9921	0.9930	0.9899	0.9920	1.0000				
sumV	0.5238	0.5300	0.9101	0.9684	0.9933	0.9958	0.9890	0.9902	0.9869	0.9893	0.9997	1.0000			
optrV	0.5055	0.5121	0.9067	0.9640	0.9943	0.9917	0.9876	0.9888	0.9887	0.9848	0.9985	0.9987	1.0000		
opndV	0.5427	0.5485	0.9112	0.9704	0.9891	0.9973	0.9876	0.9888	0.9818	0.9914	0.9980	0.9984	0.9942	1.0000	
LOC	0.5410	0.5462	0.9149	0.9738	0.9792	0.9805	0.9754	0.9759	0.9722	0.9755	0.9853	0.9852	0.9835	0.9840	1.0000

Correlation coefficients observed for st-size1 and static measures obtained by using the METRE tool were also positive but, less than 0.5000 at the 0.05 level of significance. The highest correlation coefficient was observed between modular_opndV and st-size1, with

the actual value being 0.4365. The lowest coefficient was observed between modular_n_1 and st-size1. Program volume, V, correlated well with st-size1, the correlation coefficient being 0.4117. This indicates the positive correlation between the volume metric value obtained by using METRE and the symbol table size obtained by using ST_SIZE1. The values of modularV, modular_sumV, and sumV also correlated well with st-size1, the correlation coefficients being 0.4165, 0.4023, and 0.4083, respectively. Similar correlations were observed between st-size2 and all the static measures obtained by using the METRE tool. The correlations are listed in TABLE 8.

Other significant observations made from the correlation analysis of string processing programs for static measures obtained by using METRE and dynamic measures were:

1. Correlations between the total number of operands, modularN_2, vs. st-size1 and st-size2 (correlation coefficients of 0.4277 and 0.4349, respectively) were more prominent than the correlations between the total number of operators, modularN_1, vs. st-size1 and st-size2 (correlation coefficients of 0.4066 and 0.4142, respectively).

2. Correlations between the sum of the operand volumes of modules, modular_opndV, vs. st-size1 and st-size2 (correlation coefficients of 0.4365 and 0.4415) were more prominent than the correlations between the sum of the operator volumes of modules, modular_optrV, vs. st-size1 and st-size2 (correlation coefficients of 0.3712 and 0.3788, respectively).

3. Correlations between the operand volume of a program, opndV, vs. st-size1 and st-size2 (correlation coefficients of 0.4275 and 0.4346, respectively) were more prominent than the correlations between the operator volume of a program, optrV, vs. st-size1 and st-size2 (correlation coefficients of 0.3941 and 0.4023, respectively).

4. Program volume, V, correlated more with st-size2 (correlation coefficient of 0.4192) than with st-size1 (correlation coefficient of 0.4117).

Strong, positive correlations were also observed between st-size1, st-size2, vs. static measures obtained by using MY_VOL at the 0.01 level of significance. The highest correlation coefficient was observed between modular_n and st-size2, the value being 0.5835. Both modularV and V correlated well with st-size1 and st-size2. The correlation coefficients are listed in TABLE 9.

Other significant observations made from the correlation analysis of string processing programs for static measures obtained by using MY_VOL and dynamic measures were:

1. Correlations between the sum of the number of unique operators and the number of unique operands, modular_n, vs. st-size1 and st-size2 (correlation coefficients of 0.5782 and 0.5835, respectively) were more prominent than the correlations between the sum of the number of operators and the number of operands, modularN, vs. st-size1 and st-size2 (correlation coefficients of 0.5152 and 0.5214, respectively).

2. Correlations between the program volume, V, vs. st-size1 and st-size2 (correlation coefficients of 0.5283 and 0.5344, respectively) were more prominent than the correlations between the sum of volumes of modules, modularV, vs. st-size1 and st-size2 (correlation coefficients of 0.5143 and 0.5202, respectively).

The LOC metric correlated well with both st-size1 and st-size2, correlation coefficients being 0.5410 and 0.5462 respectively (significance level of 0.01). As expected, the LOC metric correlated well with all the static measures obtained by using the three tools, the correlation coefficients being greater than 0.9000 (significance level of 0.01).

TABLE 8. Correlations between dynamic measures and static measures obtained by using METRE and CSIZE for the string processing programs

	st-size1	st-size2	modular_n_1	modular_n_2	modularN$_1$	modularN$_2$	modularV	modular_sumV	modular_optrV	modular_opndV	V	sumV	optrV	opndV	LOC
st-size1	1.0000														
st-size2	0.9984	1.0000													
modular_n_1	0.3468	0.3582	1.0000												
modular_n_2	0.4137	0.4225	0.9892	1.0000											
modularN$_1$	0.4066	0.4142	0.9597	0.9779	1.0000										
modularN$_2$	0.4277	0.4349	0.9430	0.9668	0.9971	1.0000									
modularV	0.4165	0.4226	0.9227	0.9529	0.9932	0.9960	1.0000								
modular_sumV	0.4023	0.4087	0.9212	0.9505	0.9923	0.9952	0.9996	1.0000							
modular_optrV	0.3712	0.3788	0.9384	0.9599	0.9950	0.9936	0.9964	0.9976	1.0000						
modular_opndV	0.4365	0.4415	0.8954	0.9336	0.9830	0.9909	0.9973	0.9967	0.9886	1.0000					
V	0.4117	0.4192	0.9695	0.9869	0.9983	0.9941	0.9873	0.9858	0.9894	0.9755	1.0000				
sumV	0.4083	0.4161	0.9726	0.9883	0.9977	0.9929	0.9852	0.9836	0.9880	0.9725	0.9999	1.0000			
optrV	0.3941	0.4023	0.9794	0.9910	0.9957	0.9880	0.9795	0.9780	0.9848	0.9640	0.9987	0.9992	1.0000		
opndV	0.4275	0.4346	0.9600	0.9817	0.9978	0.9972	0.9906	0.9890	0.9898	0.9821	0.9989	0.9984	0.9954	1.0000	
LOC	0.5410	0.5462	0.9178	0.9395	0.9629	0.9600	0.9542	0.9481	0.9451	0.9459	0.9599	0.9591	0.9567	0.9598	1.0000

TABLE 9. Correlations between dynamic measures and static measures obtained by using MY_VOL and CSIZE for the string processing programs

	st-size1	st-size2	modularV	modular_n	modularN	V	LOC
st-size1	1.0000						
st-size2	0.9984	1.0000					
modularV	0.5143	0.5202	1.0000				
modular_n	0.5782	0.5835	0.9241	1.0000			
modularN	0.5152	0.5214	0.9971	0.9446	1.0000		
V	0.5283	0.5344	0.9921	0.9602	0.9975	1.0000	
LOC	0.5410	0.5462	0.9781	0.9673	0.9832	0.9868	1.0000

5.2 Network Programs

For this class of programs, correlation coefficients around 0.2500 were observed between st-size1, st-size2, and all the static measures obtained by using the HALSTEAD tool. The highest correlation coefficient was observed between modular_n_1 and st-size2, the value being 0.8444 (significance level of 0.01). Other coefficients were not found to be significant even at the 0.05 level of significance. Correlation coefficients of 0.1026, 0.1115, 0.2735, and 0.3163 were observed between st_size1 vs. modularV, modular_sumV, V, and sumV, respectively, the correlation coefficient between sumV and st-size1 being the highest. Similar

insignificant correlations were observed between st-size2 vs. modularV, modular_sumV, V, and sumV, with the highest correlation coefficient of 0.3360 between modular_sumV and st-size2. Various statistical measures for static and dynamic measures for the network programs are given in TABLE 10. TABLE 11 shows the correlation coefficients for static measures obtained by using the HALSTEAD tool.

Static measures obtained by using METRE also did not correlate well with st-size1 or st-size2, the exception being modular_n_1. The highest correlation coefficient of 0.7683 (level of significance being 0.01) was observed between st-size2 and modular_n_1. Correlation coefficients between st-size1 vs. modularV, modular_sumV, V, and sumV were 0.0968, 0.0915, 0.3115, and 0.3506, respectively. These values were insignificant even at the 0.05 level of significance. Similar coefficients existed for st-size2 vs. other measures. These results are listed in TABLE 12.

For the network programs only modular_n_1 (number of operators) correlated well (significance level of 0.01) with the symbol table size measures. These insignificant correlations between the static volume metric measures (modularV, modular_sumV, V, and sumV) and the dynamic measurements indicate that the symbol table size does not vary with the volume metric for network programs. However, this result should be taken with caution because of the small sample size involved. The Pearson product-moment correlation analysis may not be applicable for this data. The standard deviation values for both the static and dynamic measures were of the same magnitude but the kurtosis value suggested that the distribution of the dynamic measures was more peaked than that of the static measures.

TABLE 10. Statistical measures for static and dynamic metrics for the 13 network programs in the test suite

Statistical measures for static measures obtained by using HALSTEAD

	$modular\_n_1$	$modular\_n_2$	$modularN_1$	$modularN_2$	modularV	modular_sumV	modular_optrV	modular_opndV	V	sumV	optrV	opndV
Minimum	18	34	131	86	1237	984	546	438	1237	984	546	438
Maximum	56	117	402	243	3951	3155	1807	1348	4795	4004	2335	1669
Mean	28.00	58.38	228.69	143.54	2327.54	1866.31	1048.92	817.38	2418.15	1957.77	1106.08	851.77
Median	23	55	212	128	2024	1608	951	716	2216	1741	951	743
Standard Deviation	11.17	21.39	81.81	49.36	888.47	714.57	401.10	316.19	1027.45	858.51	497.80	364.39
Kurtosis	2.63	4.34	-0.12	-0.47	-1.06	-1.08	-0.88	-1.29	0.84	1.27	1.85	0.47
Skewness	1.78	1.75	0.68	0.68	0.48	0.48	0.50	0.47	0.98	1.09	1.20	0.94

Statistical measures for static measures obtained by using METRE

	$modular\_n_1$	$modular\_n_2$	$modularN_1$	$modularN_2$	modularV	modular_sumV	modular_optrV	modular_opndV	V	sumV	optrV	opndV
Minimum	12	30	86	69	846	666	327	339	846	666	327	339
Maximum	39	105	271	197	2736	2133	1079	1054	3356	2755	1432	1323
Mean	19.38	51.31	155.46	116.31	1599.31	1248.54	617.38	631.23	1688.00	1339.23	670.69	668.54
Median	16	46	140	110	1330	1023	534	507	1471	1126	534	608
Standard Deviation	8.36	20.08	56.38	40.05	627.09	494.30	249.14	247.09	740.95	614.20	324.24	292.64
Kurtosis	1.23	3.64	-0.43	-0.49	-1.14	-1.17	-1.02	-1.33	0.49	0.81	1.06	0.43
Skewness	1.46	1.57	0.66	0.70	0.54	0.55	0.59	0.50	0.94	1.04	1.11	0.93

Statistical measures for dynamic measures and static measures obtained by using MY_VOL and CSIZE

	modular_n	modularN	modularV	V	LOC	st-size1	st-size2
Minimum	52	214	1220	1220	39	965	542
Maximum	172	634	3873	4708	119	3997	2224
Mean	86.15	367.38	2289.54	2386.15	66.85	1647.23	918.85
Median	78	347	1962	2148	53	1341	742
Standard Deviation	30.75	130.03	882.20	1017.72	25.66	899.54	507.83
Kurtosis	4.84	-0.39	-1.12	0.66	-0.63	3.69	3.49
Skewness	1.89	0.63	0.48	0.93	0.60	2.06	2.05

Static measures obtained with MY_VOL also did not correlate well (at significance level of 0.05) with st-size1 or st-size2. They are listed in TABLE 13.

The LOC metric correlated well with the other static measures (at the level of significance of 0.01) as expected but, it did not correlate well with the dynamic measures.

TABLE 11. Correlations between dynamic measures and static measures obtained by using HALSTEAD and CSIZE for the network programs

	st-size1	st-size2	modular_n_1	modular_n_2	modularN$_1$	modularN$_2$	modularV	modular_sumV	modular_optrV	modular_opndV	V	sumV	optrV	opndV	LOC
st-size1	1.0000														
st-size2	0.9995	1.0000													
modular_n_1	0.8331	0.8444	1.0000												
modular_n_2	0.3798	0.4012	0.8002	1.0000											
modularN$_1$	0.2106	0.2300	0.6820	0.9420	1.0000										
modularN$_2$	0.1719	0.1908	0.6519	0.9252	0.9943	1.0000									
modularV	0.1026	0.1208	0.5995	0.9033	0.9894	0.9951	1.0000								
modular_sumV	0.1115	0.1292	0.6078	0.9003	0.9870	0.9945	0.9994	1.0000							
modular_optrV	0.1742	0.1917	0.6591	0.9153	0.9914	0.9937	0.9957	0.9971	1.0000						
modular_opndV	0.0302	0.0480	0.5371	0.8736	0.9731	0.9872	0.9957	0.9953	0.9852	1.0000					
V	0.2735	0.2934	0.7336	0.9657	0.9950	0.9899	0.9778	0.9766	0.9841	0.9587	1.0000				
sumV	0.3163	0.3360	0.7656	0.9715	0.9893	0.9834	0.9678	0.9675	0.9781	0.9458	0.9987	1.0000			
optrV	0.3816	0.4010	0.8105	0.9757	0.9787	0.9674	0.9479	0.9481	0.9645	0.9193	0.9922	0.9969	1.0000		
opndV	0.2238	0.2437	0.6964	0.9559	0.9940	0.9955	0.9854	0.9843	0.9869	0.9726	0.9975	0.9942	0.9827	1.0000	
LOC	0.4356	0.4503	0.8279	0.8962	0.9427	0.9340	0.9183	0.9245	0.9481	0.8867	0.9503	0.9578	0.9662	0.9368	1.0000

TABLE 12. Correlations between dynamic measures and static measures obtained by using METRE and CSIZE for the network programs

	st-size1	st-size2	modular_n_1	modular_n_2	modularN$_1$	modularN$_2$	modularV	modular_sumV	modular_optrV	modular_opndV	V	sumV	optrV	opndV	LOC
st-size1	1.0000														
st-size2	0.9995	1.0000													
modular_n_1	0.7575	0.7683	1.0000												
modular_n_2	0.4685	0.4886	0.8656	1.0000											
modularN$_1$	0.2516	0.2701	0.7615	0.9398	1.0000										
modularN$_2$	0.2119	0.2306	0.7405	0.9306	0.9958	1.0000									
modularV	0.0968	0.1153	0.6650	0.8897	0.9862	0.9906	1.0000								
modular_sumV	0.0915	0.1097	0.6687	0.8846	0.9836	0.9884	0.9994	1.0000							
modular_optrV	0.1499	0.1676	0.7165	0.8982	0.9876	0.9866	0.9940	0.9961	1.0000						
modular_opndV	0.0327	0.0513	0.6157	0.8640	0.9720	0.9824	0.9968	0.9960	0.9842	1.0000					
V	0.3115	0.3305	0.8060	0.9646	0.9950	0.9923	0.9728	0.9705	0.9772	0.9562	1.0000				
sumV	0.3506	0.3693	0.8360	0.9701	0.9891	0.9858	0.9612	0.9599	0.9703	0.9420	0.9984	1.0000			
optrV	0.4151	0.4330	0.8781	0.9715	0.9767	0.9688	0.9382	0.9381	0.9559	0.9130	0.9902	0.9961	1.0000		
opndV	0.2759	0.2954	0.7817	0.9597	0.9938	0.9955	0.9780	0.9753	0.9773	0.9656	0.9985	0.9952	0.9827	1.0000	
LOC	0.4356	0.4503	0.8606	0.9480	0.9679	0.9544	0.9289	0.9282	0.9474	0.9019	0.9726	0.9761	0.9804	0.9625	1.0000

TABLE 13. Correlations between dynamic measures and static measures obtained by using MY_VOL and CSIZE for the network programs

	st-size1	st-size2	modularV	modular_n	modularN	V	LOC
st-size1	1.0000						
st-size2	0.9995	1.0000					
modularV	0.1065	0.1243	1.0000				
modular_n	0.5482	0.5672	0.8409	1.0000			
modularN	0.2272	0.2457	0.9903	0.8961	1.0000		
V	0.2962	0.3155	0.9753	0.9353	0.9948	1.0000	
LOC	0.4356	0.4503	0.9251	0.9153	0.9567	0.9626	1.0000

5.3 Numeric Programs

Positive correlations were observed between the symbol table size and the volume metric for this class of programs. The highest correlation coefficient of 0.6137 (significance level of 0.01) was observed between st-size2 and modular_n_2 obtained by using the HALSTEAD tool. Coefficients around 0.4822 (critical value at significance level of 0.05) were observed between st-size1 and st-size2, vs. the static measures obtained by using the HALSTEAD tool. Program volume, V, correlated well with both st-size1 and st-size2 (correlation coefficients being 0.4880 and 0.5311, respectively). This indicates that there is some positive correlation between the symbol table size and the program volume. Various statistical measures for the data under study are shown in TABLE 14. The kurtosis value indicated that the distribution of the static measures was more peaked than that of the dynamic measures. The plot of $\log_{10}$(st-size1) vs. $\log_{10}$(V) is shown in Figure 4. The coefficients obtained for the static measures collected with HALSTEAD are shown in TABLE 15.

Other significant observations from the correlation analysis of the numeric programs, for the static measures obtained by using HALSTEAD and the dynamic measures, were:

1. Correlations between modularN_1 vs. st-size1 and st-size2 (correlation coefficients of 0.5205 and 0.5616, respectively) were more prominent than the correlations between modularN_2 vs. st-size1 and st-size2 (correlation coefficients of 0.4859 and 0.5273, respectively).

2. Correlations between modular_optrV vs. st-size1 and st-size2 (correlation coefficients of

0.5107 and 0.5515, respectively) were more prominent than the correlations between modular_opndV vs. st-size1 and st-size2 (correlation coefficients of 0.4986 and 0.5363, respectively).

3. Correlation between optrV and st-size2 (correlation coefficient of 0.5212) was more prominent than the correlation between opndV and st-size2 (correlation coefficient of 0.5173).

4. Program volume, V, correlated more with st-size2 (correlation coefficient of 0.5311) than with st-size1 (correlation coefficient of 0.4880).

Similar correlations were observed between st-size1 and st-size2 vs. static measures obtained by using the METRE tool. A strong, positive correlation (correlation coefficient of 0.5462) was observed between modularV and st-size2. These correlation coefficients are listed in TABLE 16.

Other significant observations made from the correlation analysis of the numeric programs, for the static measures obtained by using METRE and the dynamic measures were:

1. Correlations between modular_opndV vs. st-size1 and st-size2 (correlation coefficients of 0.5247 and 0.5558, respectively) were more prominent than the correlations between modular_optrV vs. st-size1 and st-size2 (correlation coefficients of 0.4944 and 0.5317, respectively).

2. Correlations between program volume, V, and st-size1 and st-size2 were insignificant at the 0.05 level of significance.

All the static measures obtained by using MY_VOL correlated well with st-size1 and st-size2 at significance level of 0.05 except for program volume, V, which had insignificant

correlation with st-size1. The value of modularV was found to have good correlation with the symbol table size measures. These results are listed in TABLE 17. Other significant observations were.

1. Correlations between the number of unique tokens in a program, modular_n, vs. st-size1 and st-size2 (correlation coefficients of 0.5329 and 0.5788, respectively) were more prominent than the correlations between the total number of tokens in a program, modularN, vs. st-size1 and st-size2 (correlation coefficients of 0.4965 and 0.5384, respectively).

2. Correlation between the sum of the volumes of modules, modularV, and st-size2 (correlation coefficient of 0.5435) was more prominent than the correlation between the program volume, V, and st-size2 (correlation coefficient of 0.5220).

3. Sum of the volumes of the modules, modularV, correlated more with st-size2 (correlation coefficient of 0.5435) than with st-size1 (correlation coefficient of 0.5045).

The LOC metric correlated well with both of the dynamic measures, st-size1 and st-size2, correlation coefficients being 0.5420 and 0.5810, respectively (significance level of 0.05). As expected, the LOC metric correlated well with all other static measures.

TABLE 14. Statistical measures for static and dynamic metrics for the 17 numeric programs in the test suite

Statistical measures for static measures obtained by using HALSTEAD

	modular_n_1	modular_n_2	modularN$_1$	modularN$_2$	modularV	modular_sumV	modular_optrV	modular_opndV	V	sumV	optrV	opndV
Minimum	36	68	405	298	4710	3908	2094	1814	4710	3908	2094	1814
Maximum	712	2922	26551	17343	362998	289280	149995	139279	519149	451256	251590	199665
Mean	192.71	807.18	6553.65	4526.12	90310.59	71332.00	35559.59	35771.59	116978.00	99443.71	53047.41	46396.29
Median	139	668	3602	2172	45042	34948	18913	16033	55565	45189	25605	20414
Standard Deviation	195.50	725.29	7222.31	5029.77	102698.36	81142.07	40279.32	41051.29	143506.90	124765.29	67742.11	57230.45
Kurtosis	3.24	3.93	2.85	1.94	2.23	2.38	3.25	1.72	3.20	3.40	4.16	2.60
Skewness	1.97	1.84	1.83	1.70	1.74	1.77	1.90	1.67	1.93	1.98	2.10	1.85

Statistical measures for static measures obtained by using METRE

	modular_n_1	modular_n_2	modularN$_1$	modularN$_2$	modularV	modular_sumV	modular_optrV	modular_opndV	V	sumV	optrV	opndV
Minimum	38	83	340	250	3631	2902	1499	1302	4236	3470	1784	1685
Maximum	2925	4532	21991	14261	221999	174321	96273	78054	466359	426422	253209	173213
Mean	446.82	886.29	4823.94	3415.71	55158.47	43559.06	22388.24	21171.59	91412.06	80946.94	44961.71	35985.24
Median	162	447	1949	1396	25426	19611	9874	9737	29504	26079	14236	11843
Standard Deviation	756.96	1188.13	6002.00	4122.02	65671.90	51960.40	27383.01	24728.89	128550.83	117097.83	67797.81	49534.12
Kurtosis	7.48	5.29	3.43	2.18	1.88	1.79	2.51	1.31	3.88	4.15	5.15	2.96
Skewness	2.70	2.32	1.98	1.77	1.71	1.70	1.83	1.60	2.08	2.13	2.29	1.94

Statistical measures for dynamic measures and static measures obtained by using MY_VOL and CSIZE

	modular_n	modularN	modularV	V	LOC	st-size1	st-size2
Minimum	4554	103	681	4554	121	908	504
Maximum	369433	3616	44648	527749	8343	5087	2778
Mean	89798.65	995.88	11004.24	116267.94	1966.47	3638.47	1989.65
Median	43827	799	5715	55186	1367	4691	2530
Standard Deviation	103554.05	910.31	12344.69	144848.65	2100.07	1482.57	798.59
Kurtosis	2.46	3.84	2.67	3.45	4.71	-1.13	-1.08
Skewness	1.79	1.88	1.82	1.98	2.07	-0.72	-0.71

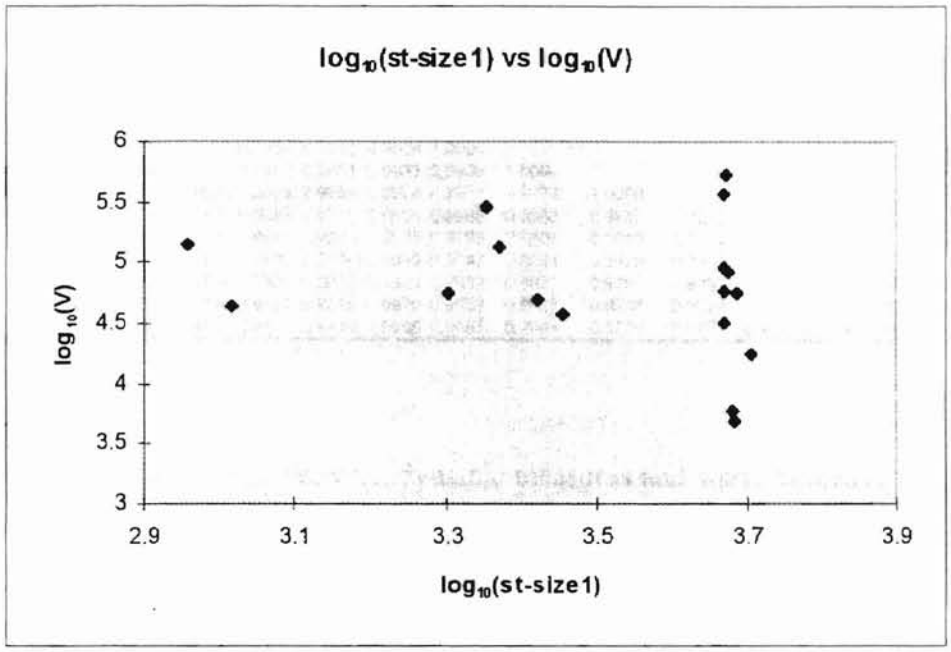

Figure 4. Plot of $\log_{10}$(st-size1) vs. $\log_{10}$(V) for the numeric programs

TABLE 15. Correlations between dynamic measures and static measures obtained by using HALSTEAD and CSIZE for the numeric programs

	st-size1	st-size2	modular_n_1	modular_n_2	modularN$_1$	modularN$_2$	modularV	modular_sumV	modular_optrV	modular_opndV	V	sumV	optrV	opndV	LOC
st-size1	1.0000														
st-size2	0.9982	1.0000													
modular_n_1	0.3797	0.4310	1.0000												
modular_n_2	0.5698	0.6137	0.9473	1.0000											
modularN$_1$	0.5205	0.5616	0.8850	0.9660	1.0000										
modularN$_2$	0.4859	0.5273	0.8853	0.9488	0.9943	1.0000									
modularV	0.5144	0.5527	0.8420	0.9403	0.9953	0.9927	1.0000								
modular_sumV	0.5058	0.5451	0.8527	0.9449	0.9968	0.9945	0.9997	1.0000							
modular_optrV	0.5107	0.5515	0.8698	0.9598	0.9988	0.9908	0.9965	0.9976	1.0000						
modular_opndV	0.4986	0.5363	0.8320	0.9258	0.9902	0.9935	0.9983	0.9977	0.9907	1.0000					
V	0.4880	0.5311	0.9029	0.9681	0.9982	0.9947	0.9906	0.9932	0.9960	0.9859	1.0000				
sumV	0.4762	0.5203	0.9140	0.9705	0.9961	0.9931	0.9860	0.9893	0.9932	0.9810	0.9995	1.0000			
optrV	0.4762	0.5212	0.9225	0.9773	0.9932	0.9857	0.9792	0.9832	0.9906	0.9713	0.9973	0.9986	1.0000		
opndV	0.4745	0.5173	0.9006	0.9589	0.9960	0.9982	0.9905	0.9930	0.9926	0.9889	0.9985	0.9980	0.9933	1.0000	
LOC	0.5420	0.5810	0.8454	0.9587	0.9822	0.9608	0.9793	0.9794	0.9873	0.9672	0.9753	0.9708	0.9726	0.9651	1.0000

TABLE 16. Correlations between dynamic measures and static measures obtained by using METRE and CSIZE for the numeric programs

	st-size1	st-size2	modular_n_1	modular_n_2	modularN$_1$	modularN$_2$	modularV	modular_sumV	modular_optrV	modular_opndV	V	sumV	optrV	opndV	LOC
st-size1	1.0000														
st-size2	0.9982	1.0000													
modular_n_1	0.2544	0.3084	1.0000												
modular_n_2	0.3719	0.4203	0.9737	1.0000											
modularN$_1$	0.4626	0.5034	0.9078	0.9771	1.0000										
modularN$_2$	0.4658	0.5052	0.8866	0.9670	0.9962	1.0000									
modularV	0.5119	0.5462	0.8337	0.9354	0.9882	0.9924	1.0000								
modular_sumV	0.5102	0.5447	0.8346	0.9362	0.9881	0.9933	0.9999	1.0000							
modular_optrV	0.4944	0.5317	0.8699	0.9570	0.9964	0.9968	0.9974	0.9974	1.0000						
modular_opndV	0.5247	0.5558	0.7905	0.9075	0.9730	0.9834	0.9965	0.9968	0.9885	1.0000					
V	0.4331	0.4757	0.9259	0.9858	0.9984	0.9935	0.9798	0.9801	0.9912	0.9618	1.0000				
sumV	0.4214	0.4648	0.9342	0.9890	0.9968	0.9910	0.9747	0.9751	0.9878	0.9552	0.9997	1.0000			
optrV	0.4052	0.4501	0.9488	0.9929	0.9925	0.9826	0.9632	0.9632	0.9796	0.9393	0.9972	0.9985	1.0000		
opndV	0.4415	0.4828	0.9098	0.9791	0.9979	0.9979	0.9859	0.9867	0.9943	0.9724	0.9984	0.9973	0.9918	1.0000	
LOC	0.5420	0.5810	0.8897	0.9541	0.9829	0.9686	0.9685	0.9664	0.9779	0.9477	0.9771	0.9750	0.9746	0.9709	1.0000

TABLE 17. Correlations between dynamic measures and static measures obtained by using MY_VOL and CSIZE for the numeric programs

	st-size1	st-size2	modularV	modular_n	modularN	V	LOC
st-size1	1.0000						
st-size2	0.9982	1.0000					
modularV	0.5045	0.5435	1.0000				
modular_n	0.5329	0.5788	0.9306	1.0000			
modularN	0.4965	0.5384	0.9957	0.9554	1.0000		
V	0.4784	0.5220	0.9908	0.9642	0.9982	1.0000	
LOC	0.5420	0.5810	0.9809	0.9423	0.9758	0.9756	1.0000

5.4 Graphics Programs

No significant correlations were observed between the static measures and the symbol table size for this class of programs. Positive correlation was observed only between modular_n_1 vs. st-size1, and modular_n_1 vs. st-size2.

Coefficients for correlations between st-size1 vs. modularV, modular_sumV, V, and sumV obtained by using HALSTEAD were -0.1353, -0.1320, -0.0694, and -0.0572, respectively. These coefficients were not significant at the 0.05 level of significance. The correlations for static measures vs. st-size2 were also not significant.

TABLE 18. Statistical measures for static and dynamic metrics for the 61 graphics programs in the test suite

Statistical measures for static measures obtained by using HALSTEAD

	modular_n_1	modular_n_2	modularN$_1$	modularN$_2$	modularV	modular_sumV	modular_optrV	modular_opndV	V	sumV	optrV	opndV
Minimum	14	21	56	49	574	457	219	215	574	457	219	215
Maximum	1696	5661	29174	23301	394498	314643	146466	168418	674036	603466	312976	290490
Mean	87.59	353.69	2129.75	1620.08	30047.52	23777.07	11220.43	12556.48	40723.85	34928.02	17814.30	17113.62
Median	23	64	184	163	2211	1796	839	967	2211	1796	839	967
Standard Deviation	236.02	887.62	5525.36	4276.76	79291.74	62765.75	29256.14	33532.96	119345.81	104668.86	53725.27	50972.23
Kurtosis	37.44	24.10	15.43	17.07	14.61	14.94	14.57	15.23	19.19	20.24	20.52	19.94
Skewness	5.78	4.64	3.85	4.04	3.78	3.82	3.76	3.86	4.28	4.39	4.41	4.37

Statistical measures for static measures obtained by using METRE

	modular_n_1	modular_n_2	modularN$_1$	modularN$_2$	modularV	modular_sumV	modular_optrV	modular_opndV	V	sumV	optrV	opndV
Minimum	5	17	36	34	318	223	81	143	334	236	93	143
Maximum	3216	5532	24487	16437	239773	181182	93626	87574	535889	489670	285299	204371
Mean	166.93	346.72	1600.23	1137.20	17303.10	13363.77	6707.85	6656.18	30375.52	26996.15	15070.90	11925.31
Median	24	63	142	125	1378	1061	462	578	1696	1357	624	747
Standard Deviation	483.56	853.18	4179.64	2862.05	44436.62	34286.49	17521.50	16802.01	87567.05	79306.07	45574.54	33803.14
Kurtosis	27.99	23.65	16.99	15.82	13.84	13.36	13.65	13.17	20.32	21.10	22.39	19.40
Skewness	4.98	4.50	3.93	3.82	3.65	3.61	3.64	3.59	4.27	4.35	4.47	4.19

Statistical measures for dynamic measures and static measures obtained by using MY_VOL and CSIZE

	modular_n	modularN	modularV	V	LOC	st-size1	st-size2
Minimum	36	111	574	574	21	2342	1303
Maximum	7355	53759	404042	690508	11769	12086	6480
Mean	440.70	3747.07	29999.79	40732.03	706.07	8777.92	4714.62
Median	90	356	2276	2276	69	9033	4845
Standard Deviation	1117.68	9897.17	79924.72	120828.92	1954.31	2062.99	1096.37
Kurtosis	26.69	16.82	15.14	19.87	20.79	2.72	2.79
Skewness	4.87	4.01	3.85	4.35	4.39	-1.54	-1.57

Similar coefficients were observed with METRE and MY_VOL, which were insignificant at the 0.05 level of significance. This indicates that there is no relationship between the static measures and the symbol table size for this class of programs. Various statistical measures are listed in TABLE 18. Large, positive values of kurtosis for the static measures indicates that the distribution was more peaked than the normal distribution. The

correlation coefficients obtained for these programs are listed in TABLES 19 through 21. The results obtained must be taken with caution because the inclusion of some programs under this class may not be accurate. There was a marked difference in the measures obtained for the Motif programs (downloaded from `ftp.uu.net`) and other programs in this class. This prompted for a further analysis of the correlations for the Motif programs, which is discussed in the following subsection.

TABLE 19. Correlations between dynamic measures and static measures obtained by using HALSTEAD and CSIZE for the graphics programs

	st-size1	st-size2	modular_n_1	modular_n_2	modularN$_1$	modularN$_2$	modularV	modular_sumV	modular_optrV	modular_opndV	V	sumV	optrV	opndV	LOC
st-size1	1.0000														
st-size2	0.9903	1.0000													
modular_n_1	0.0249	0.0411	1.0000												
modular_n_2	-0.0330	-0.0157	0.9754	1.0000											
modularN$_1$	-0.1241	-0.1081	0.8878	0.9618	1.0000										
modularN$_2$	-0.0995	-0.0830	0.9009	0.9684	0.9982	1.0000									
modularV	-0.1353	-0.1195	0.8574	0.9419	0.9975	0.9949	1.0000								
modular_sumV	-0.1320	-0.1160	0.8616	0.9451	0.9980	0.9960	0.9999	1.0000							
modular_optrV	-0.1407	-0.1249	0.8620	0.9464	0.9985	0.9952	0.9994	0.9996	1.0000						
modular_opndV	-0.1242	-0.1082	0.8606	0.9434	0.9969	0.9960	0.9996	0.9997	0.9985	1.0000					
V	-0.0694	-0.0531	0.9182	0.9771	0.9947	0.9979	0.9885	0.9901	0.9894	0.9900	1.0000				
sumV	-0.0572	-0.0407	0.9274	0.9812	0.9916	0.9961	0.9839	0.9858	0.9849	0.9858	0.9996	1.0000			
optrV	-0.0561	-0.0398	0.9328	0.9840	0.9906	0.9946	0.9816	0.9835	0.9831	0.9831	0.9990	0.9997	1.0000		
opndV	-0.0582	-0.0417	0.9212	0.9777	0.9921	0.9970	0.9858	0.9876	0.9863	0.9881	0.9996	0.9997	0.9989	1.0000	
LOC	-0.0755	-0.0592	0.9360	0.9818	0.9886	0.9922	0.9799	0.9810	0.9799	0.9813	0.9954	0.9958	0.9960	0.9950	1.0000

TABLE 20. Correlations between dynamic measures and static measures obtained by using METRE and CSIZE for the graphics programs

	st-size1	st-size2	modular_n_1	modular_n_2	modularN$_1$	modularN$_2$	modularV	modular_sumV	modular_optrV	modular_opndV	V	sumV	optrV	opndV	LOC
st-size1	1.0000														
st-size2	0.9903	1.0000													
modular_n_1	0.0284	0.0464	1.0000												
modular_n_2	-0.0275	-0.0103	0.9851	1.0000											
modularN$_1$	-0.1103	-0.0950	0.9424	0.9792	1.0000										
modularN$_2$	-0.1267	-0.1114	0.9298	0.9729	0.9990	1.0000									
modularV	-0.1635	-0.1495	0.8896	0.9461	0.9911	0.9949	1.0000								
modular_sumV	-0.1743	-0.1604	0.8812	0.9392	0.9884	0.9927	0.9997	1.0000							
modular_optrV	-0.1673	-0.1529	0.8944	0.9458	0.9916	0.9943	0.9987	0.9990	1.0000						
modular_opndV	-0.1812	-0.1677	0.8657	0.9303	0.9829	0.9890	0.9986	0.9989	0.9958	1.0000					
V	-0.0746	-0.0592	0.9556	0.9851	0.9977	0.9950	0.9829	0.9791	0.9834	0.9727	1.0000				
sumV	-0.0652	-0.0496	0.9604	0.9868	0.9964	0.9931	0.9792	0.9752	0.9800	0.9681	0.9998	1.0000			
optrV	-0.0501	-0.0343	0.9688	0.9892	0.9934	0.9886	0.9717	0.9672	0.9734	0.9587	0.9985	0.9993	1.0000		
opndV	-0.0855	-0.0702	0.9471	0.9815	0.9983	0.9971	0.9873	0.9839	0.9869	0.9787	0.9995	0.9988	0.9963	1.0000	
LOC	-0.0755	-0.0592	0.9518	0.9673	0.9815	0.9758	0.9618	0.9583	0.9658	0.9486	0.9845	0.9852	0.9864	0.9816	1.0000

TABLE 21. Correlations between dynamic measures and static measures obtained by using MY_VOL and CSIZE for the graphics programs

	st-size1	st-size2	modularV	modular_n	modularN	V	LOC
st-size1	1.0000						
st-size2	0.9903	1.0000					
modularV	-0.1295	-0.1136	1.0000				
modular_n	-0.0207	-0.0035	0.9314	1.0000			
modularN	-0.1071	-0.0907	0.9968	0.9568	1.0000		
V	-0.0640	-0.0475	0.9887	0.9702	0.9967	1.0000	
LOC	-0.0755	-0.0592	0.9826	0.9761	0.9924	0.9959	1.0000

5.5 Motif Programs

The static and dynamic measures for this class of programs were found to be parametric. The values of kurtosis and skewness suggested a nearly normal distribution of observations. The standard deviations were of the same magnitude for the static measures and the dynamic measures. Various statistical measures are listed in TABLE 22.

The correlation analysis yielded a strong, positive correlations, with correlation coefficients greater than 0.5, between the symbol table size measures and the static measures for the programs in this class. The values of modularV, modular_sumV, V, and sumV obtained with HALSTEAD correlated well with st-size1, at the significance level of 0.01. Only modular_n_1 had insignificant correlation vs. st-size1 and st-size2 at the significance level of 0.01. Correlation coefficients between modularV, modular_sumV, modular_optrV, and modular_opndV vs. st-size1 and st-size2 were the same as those between V, sumV, optrV, and opndV vs. st-size1 and st-size2. This is due to the fact that all of the programs in this class consisted of a single module and V differs from modularV only for programs with more than one module. The plot of $\log_{10}$(st-size1) vs. $\log_{10}$(V) is shown in Figure 5. The correlations for static measures obtained by using HALSTEAD are listed in TABLE 23.

Other significant observations made from the correlation analysis of the Motif

programs, for the static measures obtained by using HALSTEAD and the dynamic measures, were:

1. Correlations between the total number of operators, modularN_1, vs. st-size1 and st-size2 (correlation coefficients of 0.5704 and 0.5858, respectively) were more prominent than the correlations between the total number of operands, modularN_2, vs. st-size1 and st-size2 (correlation coefficients of 0.5490 and 0.5693, respectively).

2. Correlations between the operator volume of a program, optrV, vs. st-size1 and st-size2 (correlation coefficients of 0.5643 and 0.5785) were more prominent than the correlations between the operand volume of a program, opndV, vs. st-size1 and st-size2 (correlation coefficients of 0.5594 and 0.5783, respectively).

3. Program volume, V, correlated more with st-size2 (correlation coefficient of 0.5842) than with st-size1 (correlation coefficient of 0.5675).

TABLE 22. Statistical measures for static and dynamic metrics for the 46 Motif programs in the test suite

Statistical measures for static measures obtained by using HALSTEAD

	modular_n_1	modular_n_2	modularN$_1$	modularN$_2$	modularV	modular_sumV	modular_optrV	modular_opndV	V	sumV	optrV	opndV
Minimum	14	21	56	49	574	457	219	215	574	457	219	215
Maximum	35	170	607	521	8606	6778	2918	3860	8606	6778	2918	3860
Mean	21.48	63.54	188.74	165.57	2351.43	1890.78	859.07	1031.59	2351.43	1890.78	859.07	1031.59
Median	21	57	152	134	1766	1413	629	776	1766	1413	629	776
Standard Deviation	5.71	32.27	125.69	109.65	1820.15	1452.13	632.26	825.82	1820.15	1452.13	632.26	825.82
Kurtosis	-0.28	2.28	2.46	2.56	3.12	2.77	2.01	3.36	3.12	2.77	2.01	3.36
Skewness	0.68	1.49	1.66	1.68	1.81	1.74	1.59	1.87	1.81	1.74	1.59	1.87

Statistical measures for static measures obtained by using METRE

	modular_n_1	modular_n_2	modularN$_1$	modularN$_2$	modularV	modular_sumV	modular_optrV	modular_opndV	V	sumV	optrV	opndV
Minimum	5	17	36	34	318	223	81	143	334	236	93	143
Maximum	59	180	545	454	6447	4817	2014	2803	7793	6359	2957	3401
Mean	22.65	63.22	143.57	125.07	1458.15	1089.91	461.07	628.91	1796.11	1454.72	673.22	781.54
Median	20	51	118	103	1141	834	347	493	1343	1084	498	573
Standard Deviation	15.21	40.48	103.26	87.75	1195.07	899.82	381.76	523.13	1556.33	1304.19	623.24	685.12
Kurtosis	0.03	2.29	4.66	4.38	6.74	6.44	5.47	6.95	4.97	4.57	3.86	5.12
Skewness	0.91	1.61	1.98	1.99	2.37	2.32	2.10	2.45	2.11	2.06	1.91	2.17

Statistical measures for dynamic measures and static measures obtained by using MY_VOL and CSIZE

	modular_n	modularN	modularV	V	LOC	st-size1	st-size2
Minimum	36	111	574	574	21	8704	4666
Maximum	198	1137	8675	8675	237	11395	6095
Mean	85.11	361.50	2398.43	2398.43	72.15	9502.13	5089.24
Median	79	293	1807	1807	59	9073	4870
Standard Deviation	36.54	236.39	1831.67	1831.67	46.16	908.17	480.65
Kurtosis	1.26	2.36	2.96	2.96	3.24	-0.54	-0.52
Skewness	1.23	1.63	1.77	1.77	1.78	1.04	1.04

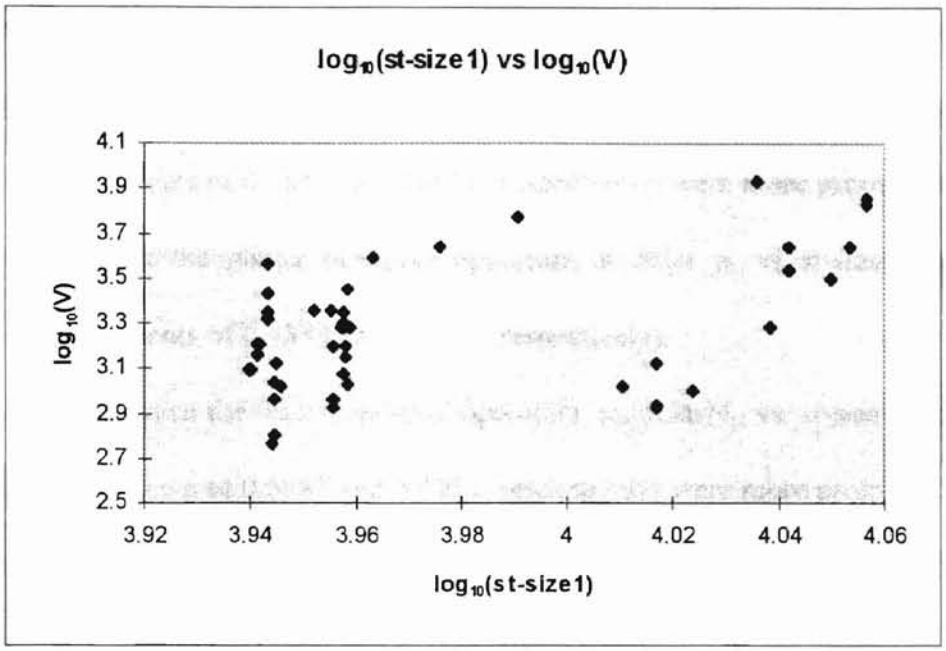

Figure 5. Plot of $\log_{10}$(st-size1) vs. $\log_{10}$(V) for the Motif programs

TABLE 23. Correlations between dynamic measures and static measures obtained by using HALSTEAD and CSIZE for the Motif programs

	st-size1	st-size2	modular_n_1	modular_n_2	modularN$_1$	modularN$_2$	modularV	modular_sumV	modular_optrV	modular_opndV	V	sumV	optrV	opndV	LOC
st-size1	1.0000														
st-size2	0.9408	1.0000													
modular_n_1	0.3088	0.3125	1.0000												
modular_n_2	0.5649	0.5948	0.7272	1.0000											
modularN$_1$	0.5704	0.5858	0.7245	0.9817	1.0000										
modularN$_2$	0.5490	0.5693	0.6851	0.9837	0.9921	1.0000									
modularV	0.5675	0.5842	0.7001	0.9855	0.9976	0.9965	1.0000								
modular_sumV	0.5638	0.5808	0.7149	0.9858	0.9980	0.9964	0.9997	1.0000							
modular_optrV	0.5643	0.5785	0.7664	0.9783	0.9972	0.9846	0.9929	0.9948	1.0000						
modular_opndV	0.5594	0.5783	0.6706	0.9846	0.9915	0.9985	0.9978	0.9969	0.9837	1.0000					
V	0.5675	0.5842	0.7001	0.9855	0.9976	0.9965	1.0000	0.9997	0.9929	0.9978	1.0000				
sumV	0.5638	0.5808	0.7149	0.9858	0.9980	0.9964	0.9997	1.0000	0.9948	0.9969	0.9997	1.0000			
optrV	0.5643	0.5785	0.7664	0.9783	0.9972	0.9846	0.9929	0.9948	1.0000	0.9837	0.9929	0.9948	1.0000		
opndV	0.5594	0.5783	0.6706	0.9846	0.9915	0.9985	0.9978	0.9969	0.9837	1.0000	0.9978	0.9969	0.9837	1.0000	
LOC	0.5268	0.5475	0.6492	0.9333	0.9557	0.9511	0.9573	0.9566	0.9510	0.9540	0.9573	0.9566	0.9510	0.9540	1.0000

More significant correlations were observed for static measures taken by METRE at the 0.01 level of significance. All the correlations, except for modular_n_1, were significant at the 0.01 level of significance (critical value for the product-moment correlation coefficient being 0.3610). The correlation coefficient for modular_n_1 was significant at the 0.05 level of

significance. These correlations are listed in TABLE 24. Other significant observations were:

1. Correlations between the number of unique operands, modular_n_2, vs. st-size1 and st-size2 (correlation coefficients of 0.5811 and 0.6068, respectively) were more prominent than the correlations between the number of unique operators, modular_n_1, vs. st-size1 and st-size2 (correlation coefficients of 0.4383 and 0.4645, respectively).

2. Correlations between the total number of operators, modularN_1, vs. st-size1 and st-size2 (correlation coefficients of 0.5887 and 0.6051, respectively) were more prominent than the correlations between the total number of operands, modularN_2, vs. st-size1 and st-size2 (correlation coefficients of 0.5715 and 0.5902, respectively).

3. Correlations between the sum of the operator volumes of modules, modular_optrV, vs. st-size1 and st-size2 (correlation coefficients of 0.5758 and 0.5881, respectively) were more prominent than the correlations between the sum of the operand volumes of modules, modular_opndV, vs. st-size1 and st-size2 (correlation coefficients of 0.5720 and 0.5868, respectively).

4. Correlations between the operator volume of a program, optrV, vs. st-size1 and st-size2 (correlation coefficients of 0.5867 and 0.6028, respectively) were more prominent than the correlations between the operand volume of a program, opndV, vs. st-size1 and st-size2 (correlation coefficients of 0.5788 and 0.5960, respectively).

5. Program volume, V, correlated more with st-size2 (correlation coefficient of 0.6028) than with st-size1 (correlation coefficient of 0.5863).

TABLE 24. Correlations between dynamic measures and static measures obtained by using METRE and CSIZE for the Motif programs

	st-size1	st-size2	modular_n_1	modular_n_2	modularN$_1$	modularN$_2$	modularV	modular_sumV	modular_optrV	modular_opndV	V	sumV	optrV	opndV	LOC
st-size1	1.0000														
st-size2	0.9408	1.0000													
modular_n_1	0.4383	0.4645	1.0000												
modular_n_2	0.5811	0.6068	0.8671	1.0000											
modularN$_1$	0.5887	0.6051	0.7935	0.9613	1.0000										
modularN$_2$	0.5715	0.5902	0.7788	0.9698	0.9949	1.0000									
modularV	0.5801	0.5943	0.7466	0.9486	0.9944	0.9937	1.0000								
modular_sumV	0.5766	0.5905	0.7624	0.9509	0.9952	0.9935	0.9994	1.0000							
modular_optrV	0.5758	0.5881	0.8116	0.9456	0.9925	0.9815	0.9882	0.9922	1.0000						
modular_opndV	0.5720	0.5868	0.7191	0.9454	0.9874	0.9926	0.9978	0.9959	0.9768	1.0000					
V	0.5863	0.6028	0.7986	0.9729	0.9966	0.9971	0.9946	0.9954	0.9884	0.9907	1.0000				
sumV	0.5843	0.6011	0.8167	0.9775	0.9950	0.9951	0.9911	0.9927	0.9882	0.9863	0.9994	1.0000			
optrV	0.5867	0.6028	0.8525	0.9753	0.9914	0.9852	0.9817	0.9853	0.9906	0.9718	0.9938	0.9964	1.0000		
opndV	0.5788	0.5960	0.7791	0.9734	0.9920	0.9980	0.9936	0.9933	0.9798	0.9935	0.9983	0.9971	0.9870	1.0000	
LOC	0.5268	0.5475	0.7970	0.9336	0.9399	0.9454	0.9371	0.9412	0.9399	0.9331	0.9483	0.9504	0.9468	0.9478	1.0000

Static measures obtained using MY_VOL also correlated well with the dynamic measures st-size1 and st-size2. The correlations were significant at the 0.01 level of significance. The coefficients obtained are listed in TABLE 25. Other significant observations were:

1. Correlation between modularN and st-size1 (correlation coefficient of 0.5550) was more prominent than the correlation between modular_n and st-size1 (correlation coefficient of 0.5434).

2. Program volume, V, correlated more with st-size2 (correlation coefficient of 0.5787) than with st-size1 (correlation coefficient of 0.5615).

TABLE 25. Correlations between dynamic measures and static measures obtained by using MY_VOL and CSIZE for the Motif programs

	st-size1	st-size2	modularV	modular_n	modularN	V	LOC
st-size1	1.0000						
st-size2	0.9408	1.0000					
modularV	0.5615	0.5787	1.0000				
modular_n	0.5434	0.5702	0.9787	1.0000			
modularN	0.5550	0.5732	0.9990	0.9791	1.0000		
V	0.5615	0.5787	1.0000	0.9787	0.9990	1.0000	
LOC	0.5268	0.5475	0.9892	0.9636	0.9886	0.9892	1.0000

The LOC metric correlated well with st-size1, st-size2, and all other static measures at the 0.01 level of significance.

CHAPTER VI

SUMMARY, CONCLUSIONS, AND FUTURE WORK

This thesis work was concerned with the investigation of the possible relationship between the static volume metric of a program as defined in Software Science and its dynamic symbol table size. Three tools, HALSTEAD, METRE, and MY_VOL, were selected to be used for measuring the volume metric. HALSTEAD and METRE were taken from the archives of the usenet newsgroup `comp.software-eng`. A few modifications were made to these tools to measure some other static measures as well (as defined in Subsection 4.2). MY_VOL was developed based on HALSTEAD. Another static metric, lines of code (LOC), was measured using the tool CSIZE which was also taken from the archives of the usenet newsgroup `comp.software-eng`. Two methods, ST_SIZE1 and ST_SIZE2, were developed based on the utilities available on the Sequent Symmetry S/81 machine in the OSU Computer Science Department.

A test suite of 100 programs obtained from various sources was utilized (see Subsection 4.4 for the program sources). After collecting the static and dynamic measurements for all the programs in the test suite, detailed correlation analysis was made. The programs in the test suite were divided into five application areas: string processing,

network, numeric, graphics, and Motif programs. The correlation analysis indicated a strong, positive correlation between Halstead's volume metric values and dynamic symbol table sizes among the string processing, numeric, and Motif programs. This is consistent with the general observation that as program size increases (with the increase in the number of operators and operands), the number of symbols in the symbol table increases also. High correlation coefficients (values greater than 0.5) between volume metric and symbol table size among the string processing, numeric, and Motif programs provide anecdotal evidence for the general observation above. Because of the differences in the variations of symbol table sizes with the variations of volume metrics among different classes of programs, the same result was not observed for the general class of programs (i.e., all programs under study). There seems to be little or no correlation between Halstead's volume metric values and the symbol table sizes for the general class of programs. However, this result should be taken with caution because of the relatively small number of programs considered for this study.

Among other correlations considered for this study, correlations between the operand volumes (modular_opndV and opndV) and symbol table sizes (st-size1 or st-size2) were more prominent than the correlations between the operator volumes (modular_optrV and optrV) and the symbol table sizes (st-size1 or st-size2) for the string processing programs. This suggests that the increase in the number of symbols in a symbol table is consistent more with the increase in the number of operands than with the increase in the number of operators for the string processing programs in the test suite. For Motif programs, correlations between the operator volumes (modular_optrV and optrV) and the symbol table sizes (st-size1 or st-size2) were more prominent than the correlations between the operand volumes

(modular_opndV and opndV) and symbol table sizes (st-size1 or st-size2) indicating that an increase in symbol table size can be attributed more to an increase in the operator volume than to an increase in the operand volume.

Since, this is an exploratory study, the results obtained may not be considered conclusive. The results only show the possibility of a relationship between the static volume metric and the dynamic symbol table size. They must be validated by conducting similar experiments on a large and more representative sample of programs and possibly for more classes of programs (i.e., application areas). Also, this study concentrated only on the operator and operand classification of the token set of programs. Other classifications of the refinements of the token set such as function calls, arithmetic operators, logical operators, constants, two-way branches, multi-way branches, etc., may throw more light on the possible correlations between static and dynamic measurements of programs. Further research may also be extended to other programming languages like C++, Pascal, Fortran etc.

REFERENCES

[Aho et al. 86] A. V. Aho, R. Sethi, and J. D. Ullman, *Compilers: Principles, Techniques, and Tools*, Addison-Wesley Publishing Company, Reading, MA, 1986.

[Albrecht 79] A. J. Albrecht, "Measuring Application Development Productivity", *Proceedings of IBM Applications Development Symposium*, pp. 83-92, GUIDE Int. and SHARE Inc., IBM Corporation, Monterey, CA, October 1979.

[Basili and Hutchens 83] V. R. Basili and D. H. Hutchens, "An Empirical Study of a Syntactic Complexity Family", *IEEE Transactions on Software Engineering*, Vol. SE-9, No. 6, pp. 664-672, November 1983.

[Bishop 87] M. Bishop, "Profiling under UNIX by Patching", *Software Practice & Experience*, Vol. 17, No. 10, pp. 729-739, October 1987.

[Bourne 78] S. R. Bourne, "UNIX Time-Sharing System: The UNIX Shell", *Bell Systems Technical Journal*, Vol. 57, No. 6, pp. 1971-1989, July-August 1978.

[Conte et al. 86] S. D. Conte, H. E. Dunsmore, and V. Y. Shen, *Software Engineering Metrics and Models*, The Benjamin/Cummings Publishing Co., Menlo Park, CA, 1986.

[Dunsmore and Gannon 79] H. E. Dunsmore and J. D. Gannon, "Data Referencing: An Empirical Investigation", *IEEE Computer*, Vol. 12, No. 12, pp. 50-59, December 1979.

[Ejiogu 91] L. O. Ejiogu, *Software Engineering with Formal Metrics*, QED Technical Publishing Group, Wellesley, MA, 1991.

[Graham et al. 83] S. L. Graham, P. B. Kessler, and M. K. McKusick, "An Execution Profiler for Modular Programs", *Software Practice & Experience*, Vol. 13, No. 8, pp. 671-685, August 1983.

[Halstead 77a] M. H. Halstead, *Elements of Software Science*, Elsevier Publishing Company, North-Holland, 1977.

[Halstead 77b] M. H. Halstead, "On Lines of Code and Programming Productivity", *IBM Systems Technical Journal*, Vol. 16, No. 4, pp. 421-423, 1977.

[Halstead 79] M. H. Halstead , "Advances in Software Science", in *Advances in Computers (Yovits, Ed.)*, Vol. 18, pp. 119-172, Academic Press, New York, NY, 1979.

[Jones 86] C. Jones, *Programming Productivity*, McGraw-Hill Book Company, NewYork, NY, 1986.

[Kafura and Henry 81] D. Kafura and S. Henry, "Software Quality Metrics Based on Interconnectivity", *The Journal of Systems and Software*, Vol. 2, No. 2, pp. 120-131, June 1981.

[Koskimies and Paakki 87] K. Koskimies and J. Paakki, "TOOLS: A Unifying Approach to Object-Oriented Language Interpretation", *ACM SIGPLAN Notices*, Vol. 22, No. 7, pp. 153-164, July 1987.

[Li and Cheung 87] H. F. Li and W. K. Cheung, "An Empirical Study of Software Metrics", *IEEE Transactions on Software Engineering*, Vol. SE-13, No. 6, pp. 697-708, June 1987.

[Mak 91] R. Mak, *Writing Compilers and Interpreters*, John Wiley & Sons, Inc., NewYork, NY, 1991.

[McCabe 76] T J McCabe, "A Complexity Measure", *IEEE Transactions on Software Engineering*, Vol. SE-2, No. 4, pp. 308-320, December 1976.

[Moll and Samadzadeh 89] K. E. Moll and M. H. Samadzadeh, "An Empirical Study of the Relationship Between Static Software Complexity Metrics and Dynamic Measurements of Pascal and C Programs", *Proceedings of the ACM South Central Regional Conference*, pp. 150-157, Tulsa, OK, November 1989.

[O'Reilly 88] T. O'Reilly, *Understanding and Using COFF*, O'Reilly & Associates, Inc., Sebastopol, CA, 1988.

[Perlis et al. 81] A. Perlis, F. Sayward, and M. Shaw, *Software Metrics: An Analysis and Evaluation*, The MIT Press, Cambridge, MA, 1981.

[Pittman and Peters 92] T. Pittman and J. Peters, *The Art of Compiler Design*, Prentice Hall, Englewood Cliffs, NJ, 1992.

[Plattner and Nievergelt 81] N. Plattner and J. Nievergelt, "Monitoring Program Execution: A Survey", *IEEE Computer*, Vol. 14, No. 11, pp. 76-93, November 1981.

[Samadzadeh and Edwards 88] M. H. Samadzadeh and W. R. Edwards, "A Classification Model of Software Comprehension", Computer Science Department, Oklahoma State University, OSU-CIS-TR-88-01, 1988.

[Samadzadeh and Nandakumar 91] M. H. Samadzadeh and C. K. Nandakumar, "A Study of Software Metrics", *Journal of Systems and Software*, Vol. 16, No. 3, pp. 229-234, November 1991.

[Shaw et al. 89] W. H. Shaw, Jr., J. W. Howatt, R. S. Maness, and D. M. Miller, "A Software Science Model of Compile Time", *IEEE Transactions on Software Engineering*, Vol. SE-15, pp. 543-549, May 1989.

[Shen et al. 83] V. Y. Shen, S. D. Conte, and H. E. Dunsmore, "Software Science Revisited: A Critical Analysis of the Theory and Its Empirical Support", *IEEE Transactions on Software Engineering*, Vol. SE-9, No. 2, pp. 155-165, March 1983.

[Tremblay and Sorenson 85] J. P. Tremblay and P. G. Sorenson, *The Theory and Practice of Compiler Writing*, McGraw-Hill Book Company, New York, NY, 1985.

[Verner and Tate 92] J. Verner and G. Tate, "A Software Size Model", *IEEE Transactions on Software Engineering*, Vol. SE-18, No. 4, pp. 265-278, April 1992.

[Wolverton 74] R. W. Wolverton, "The Cost of Developing Large-Scale Software", *IEEE Transactions on Computers*, Vol. C-23, No. 6, pp. 615-636, June 1974.

[Woodward et al. 79] M. R. Woodward, M. A. Hennell, and D. Hedley, "A Measure of Control Flow Complexity in Program Text", *IEEE Transactions on Software Engineering*, Vol. SE-5, No. 1, pp. 45-50, January 1979.

[Zweben and Fung 79] S. H. Zweben and K. C. Fung, "Exploring Software Science Relations in COBOL and APL", *Proceedings of IEEE Computer Software & Applications Conference (COMPSAC`79)*, pp. 702-707, Chicago, IL, November 1979.

APPENDICES

APPENDIX A

GLOSSARY AND TRADEMARK INFORMATION

GLOSSARY

Dynamic Measures:	Indicators of behavior of programs during execution.
LOC:	Lines Of Code metric obtained by using the CSIZE tool (as explained in Subsection 4.3.3).
modular_n:	Sum of number of unique tokens of all modules in a program.
modular_n_1:	Sum of number of unique operators of all modules in a program.
modular_n_2:	Sum of number of unique operands of all modules in a program.
modular_opndV:	Sum of operand volumes of all modules in a program.
modular_optrV:	Sum of operator volumes of all modules in a program.
modular_sumV:	Sum of operator volumes and operand volumes of all modules in a program.
modularN:	Sum of total number of tokens of all modules in a program.

modularN_1:	Sum of total number of operators of all modules in a program.
modularN_2:	Sum of total number of operands of all modules in a program.
modularV:	Sum of the volumes of all modules in program.
n:	Number of unique tokens in one module of a program.
N:	Total number of tokens in one module of a program.
n_{1m}:	Number of unique operators in one module of a program.
n_{2m}:	Number of unique operands in one module of a program.
N_{1m}:	Total number of operators in one module of a program.
N_{2m}:	Total number of operands in one module of a program.
opndV:	Operand volume of a program.
opndV_m:	Operand volume of one module of a program.
optrV:	Operator volume of a program.
optrV_m:	Operator volume of one module of a program.
Software Metrics:	Structural and syntactic measurements of software documents.
sumV:	Sum of operator volume and operand volume of a program.
Symbol Table:	A data structure used to store information relating to the symbols utilized during compilation.
V:	Volume of a program.
V_m:	Volume of one module a program.
Volume Metric:	Indicative of volume as a quantitative aspect of a program (as defined by Halstead [Halstead 79]).

TRADEMARK INFORMATION

DYNIX/ptx:	A registered trademark of Sequent Computer Systems, Inc.
Microsoft Excel:	A registered trademark of Microsoft Corporation.
Sequent S/81:	A registered trademark of Sequent Computer Systems, Inc.

APPENDIX B

SOURCE AND BRIEF DESCRIPTION OF PROGRAMS IN THE TEST SUITE

PROGRAMS FROM '/local/src' ON SEQUENT

PROGRAM	BRIEF DESCRIPTION
archie	a program that queries anonymous ftp databases
forwarder	a mail forwarding program
ghostview	a utility to view PostScript documents
man	a global man page extraction utility
rs	a reminder service program
sc	a spreadsheet calculator program
stat	a program that contains a collection of command level functions that can be interconnected to form a statistical network
utree	a screen oriented file system browser and utility
xcalendar	a calendar program with a notebook for X
xgopher	a gopher client program for X window system
xnlock	a screen lock program
xpostit	a program for manipulating on-screen post-it notes

PROGRAMS FROM '/contrib/src' ON SEQUENT

PROGRAM	BRIEF DESCRIPTION
agrep	an approximate regular expression matching utility
animate	a program that displays a sequence of images
ce	a simple unix text editor
combine	a program that combines images to create new image
convert	a program that converts images in one form to another
display	a program that displays an image on any workstation running X
expand	a program that expands tab stops
fill	a simple text formatter program
fplan	a flight planner program
friends	a program that prints which of your friends are logged on
gzip	a utility to compress files
import	a program that captures an X server screen
indent	a program that inserts or deletes spaces in a file
lander	a game program
mogrify	a program that transforms an image or a sequence of images. These transforms include image scaling, image rotation, color reduction and others
montage	a program creates a composite image by combining several separate images
othello	a game program
par	a filter program for reformatting paragraphs
sail	a game program
segment	a program that segments an image
uuconvert	a program that uudecodes a uuencoded file
xalarm	an alarm clock program for X
xasteroids	an X windows based game program
xdl	a program that displays DL animations
xgetftp	an X based tool for browsing and retrieving files from ftp sites
xtalk	a talk program between two X terminals

PROGRAMS FROM '/usr/sequent/tcp_examples'
ON SEQUENT

PROGRAM	BRIEF DESCRIPTION
tcp_client	a client program that uses tcp protocol
tcp_sock_client	a tcp client program which creates a socket and initiates connection with another socket
tcp_sock_server	a tcp server program which creates a socket and begins an infinite loop
tcp_tli_client	a tcp client program which creates an endpoint and initiates a connection with the host port
tcp_tli_server	a tcp server program which creates an endpoint and begins an infinite loop
udp_client	a client program that uses udp protocol
udp_server	a server program using udp protocol
udp_sock_client	a udp client program which sends a datagram to a receiver
udp_sock_server	a udp server program with a datagram socket
udp_tli_client	a udp client program which creates an endpoint and initiates a connection with the host port
udp_tli_server	a udp server program which creates a datagram endpoint and reads from the input

PROGRAMS FROM comp.language.c USENET
NEWSGROUP ARCHIVE

PROGRAM	BRIEF DESCRIPTION
include	a program to avoid multiple inclusion of files
opcom	a program which enables a user to execute commands as another user

PROGRAMS FROM 'ftp.uu.net'

PROGRAM	BRIEF DESCRIPTION
action_area	a motif program that shows how action area window can be used
arrow	a motif program that shows how an ArrowButton widget can be used
arrow_timer	a motif program that demonstrates continuous callbacks using ArrowButton widgets
ask_user	a motif program that uses a dialog box to post a question to user
ask_user_simple	a motif program that creates a PushButton widget and posts a dialog box that posts a question to user
attach	a motif program that demonstrates how attachments work in Form widgets
cmd_area	a motif program that uses a ScrolledText object to view the output of commands
corners	a motif program that demonstrates widget layout management for a BulletinBoard widget
dialog	a motif program that uses a dialog box
draw2	a drawing program in motif that demonstrates how to draw into an off screen pixmap
drawing	a drawing program in motif that demonstrates how to use DrawingArea widget
drawn	a motif program that uses drawn button widgets
dynapix	a motif program that demonstrates how to display a bitmap in a MainWindow
entry_cb	a motif program that demonstrates how the XmNentryCallback resource works in RowColumn widgets.
file_sel	a motif program that demonstrates how to use the XmNfileSearchProc for file selection dialog widgets.
form_corners	a motif program that demonstrates form layout management
frame	a motif program that uses a Frame widget
hello	a motif program that displays a pushbutton with label
hello_dialog	a motif program that uses an InformationDialog widget

PROGRAMS FROM 'ftp.uu.net' (cont.)

PROGRAM	BRIEF DESCRIPTION
help_text	a motif program that a creates a main window with work area and menubar
map_dlg	a motif program that demonstrates how to user XmNmapCallback
modal	a motif program that uses a modal dialog box
modify_btn	a motif program that demonstrates how a default dialog widget function can be extended
msg_area	a motif program that demonstrates how to change the bitmap and color in a MainWindow
multi_click	a motif program that demonstrates handling of multiple PushButton clicks
paned_win1	a motif program that uses PanedWindow widget
paned_win2	a motif program that uses PanedWindow widget with XtQueryGeometry
pixmaps	a motif program that uses pixmaps
proc_traversal	a motif program that demonstrates how to process keyboard traversal from a PushButton's callback routine
prompt_dlg	a motif program that demonstrates how to use PromptDialog widget
pushb	a motif program that uses a PushButton widget
radiobox	a motif program that uses a radiobox

PROGRAMS FROM 'ftp.uu.net' (cont.)

PROGRAM	BRIEF DESCRIPTION
reason	a motif program that uses reason field of the callback structure
rowcol	a motif program that uses a RowColumn widget
select_dlg	a motif program that demonstrates how to use selection boxes
show_pix	a motif program that displays a pixmap
simple_radio	a motif program that uses a radio button
spreadsheet	a spreadsheet program in motif
text_entry	a motif program that shows how the RowColumn widget can be configured to build a text entry form
tictactoe	a simple game program in motif
toggle	a motif program that uses a toggle button
toggle_box	a motif program that uses a toggle box
traversal	a motif program that demonstrates how keyboard traversal can be manipulated among primitive widgets
unit_types	a motif program that uses PanedWindow widget minimum and maximum sizes set
warning	a motif program that uses a warning dialog box
xcal	a motif program that displays a calendar

PROGRAMS WRITTEN BY THE AUTHOR IN COMPUTER NETWORKS CLASS (COMSC 4283)

PROGRAM	BRIEF DESCRIPTION
conc_serv	a program that implements concurrent server
conf_call	a program that simulates conference call
rpc	a program that implements remote procedure call

APPENDIX C

OUTPUT OF CORRELATION ANALYSIS FOR ALL PROGRAMS IN THE TEST SUITE

All the static and dynamic measures in this appendix are as defined in Subsection 4.2

DYNAMIC MEASURES ORDERED BY st-size1

PROGRAM	st-size1	st-size2
fill	781	441
expand	784	441
par	875	487
uuconvert	908	504
udp_tli_server	965	542
rs	983	551
tcp_tli_server	989	554
udp_server	1000	560
stat	1044	582
agrep	1102	638
include	1115	627
friends	1167	652
opcom	1230	690
udp_sock_server	1255	698
tcp_sock_server	1267	704
udp_tli_client	1329	736
tcp_tli_client	1341	742
indent	1356	887
tcp_client	1366	755
udp_client	1421	787
gzip	1609	1055
ce	1611	929
udp_sock_client	1662	916
tcp_sock_client	1676	923
forwarder	1925	1101
fplan	2009	1163
lander	2262	1262
archie	2285	1260
othello	2342	1303
man	2388	1509
utree	2506	1436
xdl	2643	1432
xasteroids	2856	1558
conc_serv	3146	1804
rpc	3997	2224
conf_call	4484	2510
combine	4691	2530
convert	4691	2530
import	4691	2530
mogrify	4691	2530
montage	4707	2538
segment	4753	2569
animate	4803	2581
sail	4826	2705
sc	4850	2778
display	5087	2729
xnlock	7768	4535
xtalk	7844	4222
xcalendar	7915	4262
xpostit	7946	4271

DYNAMIC MEASURES ORDERED BY st-size1 (cont.)

PROGRAM	st-size1	st-size2
xalarm	8170	4384
radiobox	8704	4666
entry_cb	8718	4673
pixmaps	8738	4683
toggle	8738	4683
frame	8747	4690
toggle_box	8777	4707
drawing	8779	4707
draw2	8780	4708
arrow_timer	8782	4708
rowcol	8798	4717
hello	8800	4718
proc_traversal	8800	4718
pushb	8800	4718
multi_click	8815	4726
simple_radio	8824	4730
corners	8960	4806
drawn	9025	4841
attach	9032	4845
form_corners	9033	4846
tictactoe	9034	5650
traversal	9069	4864
warning	9071	4869
modal	9073	4870
reason	9073	4870

PROGRAM	st-size1	st-size2
dialog	9077	4872
arrow	9083	4872
hello_dialog	9085	4876
show_pix	9088	4881
ask_user_simple	9088	4878
map_dlg	9096	4882
ask_user	9190	4939
ghostview	9208	5036
xcal	9465	5082
action_area	9787	5238
spreadsheet	10249	4747
xgopher	10305	5634
paned_win1	10401	5559
paned_win2	10401	5559
unit_types	10401	5559
text_entry	10568	5476
help_text	10867	5813
prompt_dlg	10931	5842
select_dlg	11014	5890
modify_btn	11019	5887
cmd_area	11217	6000
file_sel	11312	6051
dynapix	11394	6094
msg_area	11395	6095
xgetftp	12086	6480

STATIC MEASURES OBTAINED BY USING THE HALSTEAD TOOL

PROGRAM	modular_n_1 / modular_n_2		modularN_1 / modularN_2		modularV / modular_sumV		modular_optrV / modular_opndV		V / sumV		optrV / opndV	
action_area	32	119	416	395	5870	4803	2080	2723	5870	4803	2080	2723
agrep	445	1109	13339	9780	170454	138946	71051	67895	245102	216277	117352	98925
animate	171	1024	8473	5399	120469	93097	46883	46214	141811	116841	62851	53990
archie	536	1490	7926	5513	96447	78179	40676	37506	147620	129971	71858	58113
arrow	16	40	133	111	1417	1123	532	591	1417	1123	532	591
arrow_timer	27	73	210	196	2697	2212	999	1213	2697	2212	999	1213
ask_user	28	92	303	264	3916	3179	1457	1722	3916	3179	1457	1722
ask_user_simple	18	40	102	80	1066	851	425	426	1066	851	425	426
attach	15	30	82	72	846	674	320	353	846	674	320	353
ce	246	1080	11551	7971	158220	125876	63472	62404	202499	172066	91744	80322
cmd_area	25	89	252	210	3157	2530	1170	1360	3157	2530	1170	1360
combine	134	634	2912	1709	34574	26662	14370	12290	44292	36484	20576	15908
conc_serv	56	117	402	243	3951	3155	1807	1348	4795	4004	2335	1669
conf_call	97	336	2491	1479	26918	21142	11637	9506	34770	28853	16440	12412
convert	138	668	3602	2087	43399	33610	18276	15333	54925	45189	25605	19584
corners	21	61	191	163	2251	1806	839	967	2251	1806	839	967
dialog	17	57	158	153	1931	1538	646	892	1931	1538	646	892
display	161	1219	16045	11621	258312	198191	90750	107440	288569	236757	117625	119132
draw2	22	80	285	264	3663	2940	1271	1669	3663	2940	1271	1669
drawing	20	72	174	165	2211	1770	752	1018	2211	1770	752	1018
drawn	21	64	184	169	2263	1822	808	1014	2263	1822	808	1014
dynapix	26	139	487	433	6777	5372	2289	3083	6777	5372	2289	3083
entry_cb	20	42	110	97	1233	998	475	523	1233	998	475	523

STATIC MEASURES OBTAINED BY USING THE HALSTEAD TOOL (cont.)

PROGRAM	modular_n_1	modular_n_2	modularN_1	modularN_2	modularV	modular_sumV	modular_optrV	modular_opndV	V	sumV	optrV	opndV
expand	40	40	254	154	2579	2171	1352	820	2579	2171	1352	820
file_sel	35	96	356	258	4319	3525	1826	1699	4319	3525	1826	1699
fill	45	103	955	647	11550	9571	5245	4326	11550	9571	5245	4326
form_corners	14	29	93	74	906	714	354	359	906	714	354	359
forwarder	46	106	493	356	5368	4277	2206	2072	6153	5118	2723	2395
fplan	360	1029	7274	5632	95287	77478	37933	39547	134736	118129	61770	56360
frame	17	40	152	126	1622	1292	621	671	1622	1292	621	671
friends	32	40	145	91	1456	1209	725	484	1456	1209	725	484
ghostview	695	3674	26676	20796	394285	314643	146466	168178	574083	498134	251845	246290
gzip	688	2522	15352	12517	223297	182405	86330	96075	324628	286159	144712	141447
hello	15	32	56	58	633	509	219	290	633	509	219	290
hello_dialog	17	52	131	126	1570	1254	535	718	1570	1254	535	718
help_text	28	170	607	521	8606	6778	2918	3860	8606	6778	2918	3860
import	132	645	3186	1876	38029	29263	15687	13576	48604	39952	22443	17509
include	35	58	347	215	3568	2923	1704	1218	3675	3039	1780	1259
indent	410	1553	12155	8436	164000	132220	67834	64382	225242	194928	105499	89429
lander	177	489	2157	1729	28215	23232	11450	11782	36448	31554	16108	15446
man	201	367	1630	1109	17142	13968	7717	6250	25061	21919	12471	9448
map_dlg	22	63	153	145	1910	1549	682	867	1910	1549	682	867
modal	21	62	161	151	1989	1606	707	899	1989	1606	707	899
modify_btn	23	82	264	249	3444	2777	1194	1583	3444	2777	1194	1583
mogrify	139	675	3707	2172	45042	34948	18913	16033	56843	46804	26390	20414
montage	153	787	5657	3501	73409	57222	29848	27373	90449	74735	41055	33680
msg_area	25	146	510	458	7180	5661	2368	3293	7180	5661	2368	3293
multi_click	20	49	113	103	1319	1067	488	578	1319	1067	488	578

STATIC MEASURES OBTAINED BY USING THE HALSTEAD TOOL (cont.)

PROGRAM	modular_n_1	modular_n_2	modularN$_1$	modularN$_2$	modularV	modular_sumV	modular_optrV	modular_opndV	V	sumV	optrV	opndV
opcom	74	155	1021	649	11989	9707	5342	4364	13091	11062	6340	4722
othello	47	146	2420	1774	31843	26197	13442	12755	31843	26197	13442	12755
paned_win1	14	36	80	67	830	651	305	346	830	651	305	346
paned_win2	14	46	119	106	1329	1039	453	585	1329	1039	453	585
par	216	563	5051	3769	66209	54168	27725	26444	84720	73607	39170	34437
pixmaps	29	50	129	100	1444	1191	627	564	1444	1191	627	564
proc_traversal	15	33	84	79	910	727	328	399	910	727	328	399
prompt_dlg	18	62	151	151	1909	1529	630	899	1909	1529	630	899
pushb	19	41	98	88	1099	888	416	471	1099	888	416	471
radiobox	19	42	109	98	1228	991	463	528	1228	991	463	528
reason	25	64	184	157	2208	1796	854	942	2208	1796	854	942
rowcol	15	21	62	49	574	457	242	215	574	457	242	215
rpc	46	56	212	128	1935	1574	956	617	2269	1914	1171	743
rs	38	113	856	538	10090	8161	4492	3669	10090	8161	4492	3669
sail	624	2011	18682	14185	251910	201483	97378	104103	373487	329132	173470	155662
sc	712	2922	26551	17343	362998	289280	149995	139279	519149	451256	251590	199665
segment	152	725	5186	3401	68273	54372	28295	26076	83950	69904	37588	32316
select_dlg	33	95	337	284	4347	3566	1700	1866	4347	3566	1700	1866
show_pix	29	77	228	194	2839	2323	1108	1216	2839	2323	1108	1216
simple_radio	17	37	93	87	1036	833	380	453	1036	833	380	453
spreadsheet	23	37	95	80	1034	846	430	417	1034	846	430	417
stat	39	104	483	339	5885	4824	2553	2271	5885	4824	2553	2271
tcp_client	31	65	268	163	2838	2309	1328	982	2838	2309	1328	982
tcp_sock_client	21	41	134	90	1334	1071	589	482	1334	1071	589	482
tcp_sock_server	24	41	162	103	1596	1295	743	552	1596	1295	743	552

STATIC MEASURES OBTAINED BY USING THE HALSTEAD TOOL (cont.)

PROGRAM	modular_n_1	modular_n_2	modularN$_1$	modularN$_2$	modularV	modular_sumV	modular_optrV	modular_opndV	V	sumV	optrV	opndV
tcp_tli_client	22	53	200	125	2024	1608	892	716	2024	1608	892	716
tcp_tli_server	23	62	300	183	3096	2447	1357	1090	3096	2447	1357	1090
text_entry	17	39	99	74	1005	796	405	391	1005	796	405	391
tictactoe	22	58	129	122	1587	1290	575	715	1587	1290	575	715
toggle	20	57	134	123	1611	1297	579	717	1611	1297	579	717
toggle_box	25	54	186	145	2087	1698	864	834	2087	1698	864	834
traversal	23	67	153	141	1909	1547	692	855	1909	1547	692	855
udp_client	32	70	309	207	3443	2814	1545	1269	3443	2814	1545	1269
udp_server	27	73	298	187	3222	2574	1417	1157	3222	2574	1417	1157
udp_sock_client	21	39	146	94	1418	1138	641	497	1418	1138	641	497
udp_sock_server	18	34	131	86	1237	984	546	438	1237	984	546	438
udp_tli_client	23	55	191	119	1948	1552	864	688	1948	1552	864	688
udp_tli_server	20	53	220	138	2216	1741	951	790	2216	1741	951	790
unit_types	14	37	82	69	857	672	312	359	857	672	312	359
utree	897	3946	31440	20860	427183	337901	173270	164630	640240	557591	308394	249197
uuconvert	36	68	405	298	4710	3908	2094	1814	4710	3908	2094	1814
warning	18	48	104	94	1197	959	434	525	1197	959	434	525
xalarm	354	1449	8264	5563	106147	84708	43759	40946	149555	128392	69976	58416
xasteroids	54	361	3468	2921	55565	44774	19958	24816	55565	44774	19958	24816
xcal	34	103	343	267	4330	3530	1745	1785	4330	3530	1745	1785
xcalendar	221	986	5508	4557	81300	65560	30017	35543	103038	88217	42896	45321
xdl	47	215	1204	957	17360	14103	6688	7415	17360	14103	6688	7415
xgetftp	309	1326	7987	6057	109055	85710	40219	45491	149921	128893	66064	62828
xgopher	1696	5661	29174	23301	394498	314025	145608	168418	674036	603466	312976	290490
xnlock	70	398	2528	1918	38101	30271	14243	16028	39438	32060	15495	16565
xpostit	234	747	2597	2158	34577	27895	12958	14937	47256	41037	20439	20598
xtalk	30	167	791	602	10618	8326	3881	4445	10618	8326	3881	4445

STATIC MEASURES OBTAINED BY USING THE METRE TOOL

PROGRAM	modular_n_1	modular_n_2	modularN$_1$	modularN$_2$	modularV	modular_sumV	modular_optrV	modular_opndV	V	sumV	optrV	opndV
action_area	59	152	282	263	2965	2266	964	1303	4208	3565	1659	1906
agrep	1160	1763	12620	8877	131499	106568	57983	48592	247500	224198	128470	95728
animate	162	834	6167	4397	84944	66306	31818	34487	105217	87933	45265	42668
archie	542	972	5438	3631	56977	45436	24658	20779	95806	85426	49389	36037
arrow	6	38	113	107	1201	854	292	562	1201	854	292	562
arrow_timer	32	79	154	141	1588	1194	495	698	2004	1659	770	889
ask_user	38	92	202	181	1999	1524	660	864	2690	2241	1060	1181
ask_user_simple	12	33	78	66	791	613	280	333	791	613	280	333
attach	5	27	60	58	590	415	139	276	590	415	139	276
ce	1201	1715	10945	7231	122498	95900	50883	45019	209202	189657	111968	77690
cmd_area	23	83	231	180	2422	1752	746	1007	2765	2192	1045	1148
combine	42	191	1204	805	14949	11631	5857	5774	15799	12592	6492	6100
conc_serv	39	105	271	197	2736	2133	1079	1054	3356	2755	1432	1323
conf_call	265	443	1464	1072	14884	11733	6088	5647	24010	21209	11785	9424
convert	44	302	1860	1222	24087	18382	9087	9295	25996	20222	10155	10067
corners	21	51	142	126	1378	1061	475	586	1654	1338	624	715
dialog	24	62	135	110	1180	839	356	483	1574	1274	619	655
display	543	1888	13845	10236	181500	143389	69791	73600	270847	237174	125779	111395
draw2	19	75	244	216	2555	1908	782	1124	3015	2382	1036	1345
drawing	20	63	141	125	1427	1082	465	618	1696	1357	609	747
drawn	19	69	146	133	1526	1141	459	682	1802	1433	620	812
dynapix	53	168	381	339	4281	3199	1286	1913	5607	4688	2182	2506
entry_cb	21	35	70	60	635	492	241	251	755	615	307	308

STATIC MEASURES OBTAINED BY USING THE METRE TOOL (cont.)

PROGRAM	modular_n_1	modular_n_2	modularN$_1$	modularN$_2$	modularV	modular_sumV	modular_optrV	modular_opndV	V	sumV	optrV	opndV
expand	52	42	475	210	3897	3239	2286	953	4490	3840	2708	1132
file_sel	58	94	255	186	2351	1858	991	867	3196	2713	1494	1219
fill	91	124	1059	622	10566	8621	5141	3480	13025	11217	6892	4326
form_corners	6	22	62	60	586	428	160	268	586	428	160	268
forwarder	92	144	401	293	3563	2729	1406	1323	5471	4717	2616	2101
fplan	764	1127	5044	3641	48739	38365	20389	17979	94536	85222	48309	36913
frame	5	33	83	81	861	601	193	409	861	601	193	409
friends	22	32	81	61	817	666	361	305	817	666	361	305
ghostview	1683	2728	15187	10226	156975	123295	65782	57512	307672	279472	162756	116716
gzip	1659	2205	11952	7829	116348	94092	54202	39887	235708	214793	127840	86953
hello	7	23	36	34	318	223	81	143	343	255	101	154
hello_dialog	15	49	92	82	841	608	241	367	1044	820	359	460
help_text	43	180	545	454	6447	4817	2014	2803	7793	6359	2957	3401
import	41	233	1469	1012	18966	14569	7049	7520	20091	15829	7870	7959
include	67	96	263	179	2167	1662	916	746	3248	2774	1595	1179
indent	550	1178	9275	6213	110892	88139	46289	41851	166572	147819	84433	63386
lander	289	493	1727	1242	16474	13051	6975	6074	28535	25228	14118	11110
man	138	291	1003	751	9719	7632	3887	3746	15338	13277	7130	6147
map_dlg	23	57	130	103	1115	827	386	441	1473	1189	588	601
modal	17	59	132	113	1254	921	384	537	1531	1204	540	665
modify_btn	34	101	230	197	2209	1613	672	942	3022	2482	1170	1312
mogrify	47	275	1949	1267	25426	19611	9874	9737	26792	21093	10826	10267
montage	94	447	3614	2565	48298	37959	18612	19347	56102	46271	23688	22583
msg_area	52	179	402	359	4588	3420	1355	2064	5975	4978	2292	2687
multi_click	20	45	75	67	660	479	199	279	855	692	324	368

STATIC MEASURES OBTAINED BY USING THE METRE TOOL (cont.)

PROGRAM	modular_n_1	modular_n_2	modularN_1	modularN_2	modularV	modular_sumV	modular_optrV	modular_opndV	V	sumV	optrV	opndV
opcom	166	196	1003	561	8275	6658	4199	2459	13294	11669	7397	4272
othello	236	358	1806	1396	17991	14452	7685	6766	29504	26079	14236	11843
paned_win1	5	32	57	55	583	407	132	275	583	407	132	275
paned_win2	7	43	101	95	1106	799	284	515	1106	799	284	515
par	561	754	4708	3070	47944	38841	22290	16556	80587	72337	42993	29344
pixmaps	26	43	99	79	948	735	366	369	1087	894	465	429
proc_traversal	8	26	56	52	504	353	127	227	549	412	168	244
prompt_dlg	18	62	127	112	1167	837	332	506	1511	1196	530	667
pushb	14	36	71	65	636	463	188	275	768	606	270	336
radiobox	15	41	78	74	743	536	202	334	883	701	305	396
reason	29	72	146	122	1365	1028	462	565	1784	1462	709	753
rowcol	5	17	40	35	334	236	93	143	334	236	93	143
rpc	31	58	156	111	1330	1023	541	483	1729	1423	773	650
rs	140	213	727	509	7265	5810	3136	2673	10461	9120	5183	3937
sail	1667	2898	13578	10191	147850	119464	62037	57431	288945	262531	145326	117205
sc	2925	4532	21991	14261	221999	174321	96273	78054	466359	426422	253209	173213
segment	277	456	3063	2180	32675	26192	14035	12160	49901	44108	24852	19256
select_dlg	40	106	225	199	2221	1698	749	950	3048	2536	1197	1339
show_pix	37	76	153	125	1416	1077	498	578	1896	1578	797	781
simple_radio	13	35	68	66	631	453	167	287	748	590	252	339
spreadsheet	14	28	66	57	663	525	251	274	663	525	251	274
stat	38	107	340	250	3924	3070	1499	1570	4236	3470	1784	1685
tcp_client	20	58	182	132	1974	1560	787	773	1974	1560	787	773
tcp_sock_client	14	30	86	69	846	666	327	339	846	666	327	339
tcp_sock_server	18	40	128	91	1283	1018	534	484	1283	1018	534	484

STATIC MEASURES OBTAINED BY USING THE METRE TOOL (cont.)

PROGRAM	modular_n_1	modular_n_2	modularN_1	modularN_2	modularV	modular_sumV	modular_optrV	modular_opndV	V	sumV	optrV	opndV
tcp_tli_client	14	42	120	94	1243	964	457	507	1243	964	457	507
tcp_tli_server	16	58	209	154	2254	1738	836	902	2254	1738	836	902
text_entry	12	30	75	58	717	553	269	285	717	553	269	285
tictactoe	22	50	99	89	987	760	337	422	1160	944	441	502
toggle	17	51	104	98	1108	831	328	503	1230	981	425	556
toggle_box	36	53	128	102	1186	921	459	461	1489	1246	662	584
traversal	24	58	122	107	1229	953	432	522	1456	1186	559	627
udp_client	29	65	219	168	2403	1919	956	963	2537	2076	1064	1012
udp_server	18	59	199	146	2162	1689	830	859	2162	1689	830	859
udp_sock_client	14	30	97	75	939	737	369	368	939	737	369	368
udp_sock_server	12	32	100	76	961	738	358	380	961	738	358	380
udp_tli_client	14	44	114	89	1189	920	434	486	1189	920	434	486
udp_tli_server	13	46	140	110	1471	1126	518	608	1471	1126	518	608
unit_types	5	33	59	57	609	425	137	288	609	425	137	288
utree	4100	6633	27953	17311	271347	216146	123720	92411	606074	555246	335475	219771
uuconvert	63	83	392	274	3631	2902	1600	1302	4788	4090	2343	1747
warning	12	43	72	67	671	489	189	299	804	622	258	364
xalarm	914	1624	7061	4647	68245	54188	30052	24132	132411	119014	69452	49562
xasteroids	278	598	2754	2262	31699	25239	12169	13070	49030	43224	22360	20865
xcal	51	104	237	198	2483	1922	891	1032	3165	2671	1344	1327
xcalendar	403	773	2838	2154	28963	22431	11122	11310	50917	45228	24562	20666
xdl	86	245	1204	866	14542	11601	5850	5751	17327	14610	7737	6873
xgetftp	954	1988	7628	5421	74452	53933	25337	28593	150358	134899	75501	59398
xgopher	3216	5532	24487	16437	239773	181182	93626	87574	535889	489670	285299	204371
xnlock	253	551	2015	1551	21666	17342	8760	8583	34416	30209	16086	14123
xpostit	286	542	2481	1601	23454	17710	9362	8349	39569	34785	20245	14541
xtalk	44	157	864	605	10295	7786	3710	4076	11239	9130	4717	4413

STATIC MEASURES OBTAINED BY USING THE MY_VOL AND CSIZE TOOLS

PROGRAM	modular_n	modularN	modularV	V	LOC
action_area	151	837	6059	6059	168
agrep	1788	23828	175228	257441	3465
animate	1186	13382	116395	136655	2533
archie	2024	13818	99287	151763	2918
arrow	56	248	1440	1440	59
arrow_timer	101	423	2816	2816	82
ask_user	121	592	4096	4096	121
ask_user_simple	107	442	2980	2980	101
attach	45	157	862	862	34
ce	1317	19717	159795	204328	4238
cmd_area	114	464	3170	3170	78
combine	760	4454	33316	42624	1067
conc_serv	172	634	3873	4708	119
conf_call	427	3810	25765	33292	698
convert	799	5534	42253	53359	1367
corners	82	361	2295	2295	76
dialog	74	323	2006	2006	66
display	1375	27354	255843	285171	4898
draw2	102	546	3643	3643	83
drawing	92	347	2264	2264	59
drawn	84	356	2276	2276	65
dynapix	164	917	6747	6747	174
entry_cb	62	218	1298	1298	46
expand	80	429	2712	2712	106
file_sel	132	613	4318	4318	119
fill	148	1589	11456	11456	324
form_corners	43	166	901	901	37
forwarder	153	872	5514	6328	189
fplan	1381	13095	96905	136600	2192
frame	57	281	1639	1639	77
friends	73	237	1467	1467	51
ghostview	4357	47920	397381	579311	8475
gzip	3147	27083	216236	314698	5220
hello	47	121	672	672	29
hello_dialog	69	268	1637	1637	53
help_text	198	1137	8675	8675	237
import	769	4873	36606	46717	1131
include	86	561	3605	3605	113
indent	1949	21769	173151	237903	4660

STATIC MEASURES OBTAINED BY USING THE MY_VOL AND CSIZE TOOLS
(cont.)

PROGRAM	modular_n	modularN	modularV	V	LOC
lander	665	3870	28148	36290	825
man	566	2752	17218	25166	626
map_dlg	85	306	1961	1961	61
modal	83	319	2034	2034	64
modify_btn	105	524	3518	3518	100
mogrify	807	5715	43827	55186	1374
montage	931	8905	71510	87827	1805
msg_area	170	966	7157	7157	185
multi_click	69	224	1368	1368	53
opcom	230	1656	11179	12992	311
othello	192	4039	30636	30636	560
paned_win1	50	149	841	841	32
paned_win2	60	230	1359	1359	44
par	783	9112	68437	87592	1430
pixmaps	78	234	1471	1471	49
proc_traversal	48	167	933	933	34
prompt_dlg	80	310	1960	1960	57
pushb	60	188	1110	1110	38
radiobox	62	215	1280	1280	40
reason	90	358	2324	2324	65
rowcol	36	111	574	574	21
rpc	100	353	4802	2345	83
rs	152	1380	10002	1002	343
sail	2644	32715	250657	371921	4147
sc	3616	44648	369433	527749	8343
segment	874	8290	66103	81006	1749
select_dlg	128	646	4522	4522	128
show_pix	58	178	1043	1043	37
simple_radio	55	187	1081	1081	38
spreadsheet	60	181	1069	1069	34
stat	141	822	6000	5869	155
tcp_client	96	433	2851	2851	86
tcp_sock_client	62	219	1304	1304	40
tcp_sock_server	65	265	1596	1596	52
tcp_tli_client	75	315	1962	1962	53
tcp_tli_server	85	474	3038	3038	83
text_entry	56	177	1028	1028	37
tictactoe	80	261	1650	1650	53
toggle	78	263	1653	1653	54

STATIC MEASURES OBTAINED BY USING THE MY_VOL AND CSIZE TOOLS
(cont.)

PROGRAM	modular_n	modularN	modularV	V	LOC
toggle_box	80	337	2130	2130	69
traversal	90	305	1980	1980	64
udp_client	102	517	3450	3450	95
udp_server	100	474	3149	3149	81
udp_sock_client	60	232	1370	1370	39
udp_sock_server	52	214	1220	1220	39
udp_tli_client	78	299	1879	1879	47
udp_tli_server	73	347	2148	2148	52
unit_types	51	153	868	868	33
utree	4824	51228	418341	626827	9413
uuconvert	103	681	4554	4554	121
warning	66	204	1233	1233	43
xalarm	1793	13000	99568	140506	1869
xasteroids	426	6493	56714	56714	792
xcal	136	619	4387	4387	122
xcalendar	1211	10147	81858	103925	1722
xdl	261	2202	17677	17677	371
xgetftp	1632	13716	106283	146383	2653
xgopher	7355	53759	404042	690508	11769
xnlock	470	4428	37964	39305	708
xpostit	985	4797	34883	47701	973
xtalk	196	1378	10493	10493	230

Statistical measures for static and dynamic measures for all 100 programs in the test suite

Statistical measures for static measures obtained by using the HALSTEAD and CSIZE tools

	moduar_n_2		modularN$_2$		modular_sumV		modular_opndV		sumV		opndV
	modular_n_1	modularN$_1$		modularV		modular_optrV		V		optrV	
Minimum	14	21	56	49	574	457	219	215	574	457	219	215
Maximum	1696	5661	31440	23301	427183	337901	173270	168418	674036	603466	312976	290490
Mean	122.83	453.93	3115.98	2254.62	42387.55	33778.44	16655.95	17122.27	58546.49	50531.97	26695.71	23836.21
Median	29	75	299	213	3444	2796	1387	1309	3444	2796	1387	1315
Standard Deviation	237.01	898.25	6450.61	4704.21	89910.15	71513.79	34976.57	36642.18	133389.81	116741.78	61939.40	54925.11
Kurtosis	20.50	14.37	8.40	8.58	8.29	8.24	8.39	8.37	10.29	10.64	10.79	10.71
Skewness	3.99	3.50	2.90	2.93	2.91	2.90	2.90	2.93	3.18	3.23	3.25	3.23

Statistical measures for static measures obtained by using the METRE and CSIZE tools

	moduar_n_2		modularN$_2$		modular_sumV		modular_opndV		sumV		opndV
	modular_n_1	modularN$_1$		modularV		modular_optrV		V		optrV	
Minimum	5	17	36	34	318	223	81	143	334	236	93	143
Maximum	4100	6633	27953	17311	271347	216146	123720	92411	606074	555246	335475	219771
Mean	274.49	493.38	2425.60	1663.94	25680.41	20186.10	10650.31	9536.09	46526.44	41707.97	23888.89	17819.13
Median	37	81	228	181	2238	1718	809	922	2890	2312	1062	1164
Standard Deviation	664.54	1082.32	5212.68	3461.76	53857.58	42367.91	22851.03	19607.83	109654.74	99878.12	58626.71	41345.00
Kurtosis	16.27	15.81	9.94	8.50	8.03	7.92	8.99	7.11	11.85	12.19	13.17	10.92
Skewness	3.84	3.76	3.07	2.90	2.84	2.83	2.95	2.73	3.34	3.38	3.49	3.23

Statistical measures obtained by using the MY_VOL and CSIZE tools and dynamic measures obtained by using the ST_SIZE1 and ST_SIZE2 tools

	modular_n	modularN	modularV	V	LOC	st-size1	st-size2
Minimum	36	111	574	574	21	781	441
Maximum	7355	53759	418341	690508	11769	12086	6480
Mean	577.11	5374.63	42414.12	58546.07	1001.22	6178.09	3339.12
Median	102.50	520.50	3561.50	3310.00	100.50	8058	4460
Standard Deviation	1129.16	11192.71	90174.66	134123.77	2110.94	3742.45	1985.70
Kurtosis	15.43	8.44	8.32	10.38	10.70	-1.60	-1.59
Skewness	3.58	2.92	2.91	3.19	3.16	-0.22	-0.24

Correlations between dynamic and static measures obtained by using the HALSTEAD and CSIZE tools for all programs in the test suite

	st_size1	st_size2	modular_n_1	modular_n_2	modularN$_1$	modularN$_2$	modularV	modular_sumV	modular_optrV	modular_opndV	V	sumV	optrV	opndV	LOC
st-size1	1.0000														
st-size2	0.9981	1.0000													
modular_n1	-0.1101	-0.0976	1.0000												
modular_n2	-0.0801	-0.0675	0.9745	1.0000											
modularN1	-0.1440	-0.1318	0.9088	0.9589	1.0000										
modularN2	-0.1252	-0.1128	0.9253	0.9687	0.9954	1.0000									
modularV	-0.1339	-0.1218	0.8939	0.9525	0.9980	0.9953	1.0000								
modular_sumV	-0.1362	-0.1240	0.8970	0.9538	0.9983	0.9961	0.9999	1.0000							
modular_optrV	-0.1511	-0.1389	0.8916	0.9481	0.9988	0.9919	0.9983	0.9985	1.0000						
modular_opndV	-0.1215	-0.1093	0.8996	0.9565	0.9950	0.9973	0.9986	0.9986	0.9941	1.0000					
V	-0.1131	-0.1009	0.9293	0.9733	0.9957	0.9968	0.9939	0.9945	0.9922	0.9938	1.0000				
sumV	-0.1104	-0.0981	0.9351	0.9756	0.9939	0.9961	0.9914	0.9922	0.9897	0.9917	0.9998	1.0000			
optrV	-0.1181	-0.1058	0.9328	0.9724	0.9945	0.9931	0.9903	0.9911	0.9909	0.9884	0.9989	0.9991	1.0000		
opndV	-0.1016	-0.0892	0.9357	0.9770	0.9909	0.9972	0.9904	0.9912	0.9862	0.9932	0.9986	0.9988	0.9958	1.0000	
LOC	-0.1246	-0.1122	0.9381	0.9786	0.9856	0.9865	0.9832	0.9835	0.9811	0.9830	0.9893	0.9889	0.9877	0.9881	1.0000

Correlations between dynamic and static measures obtained by using the METRE and CSIZE tools for all programs in the test suite

	st_size1	st_size2	modular_n_1	modular_n_2	modularN$_1$	modularN$_2$	modularV	modular_sumV	modular_optrV	modular_opndV	V	sumV	optrV	opndV	LOC
st-size1	1.0000														
st-size2	0.9981	1.0000													
modular_n1	-0.1236	-0.1114	1.0000												
modular_n2	-0.1030	-0.0911	0.9900	1.0000											
modularN1	-0.1531	-0.1415	0.9641	0.9804	1.0000										
modularN2	-0.1498	-0.1381	0.9514	0.9725	0.9979	1.0000									
modularV	-0.1632	-0.1520	0.9331	0.9594	0.9946	0.9966	1.0000								
modular_sumV	-0.1712	-0.1600	0.9318	0.9571	0.9938	0.9958	0.9997	1.0000							
modular_optrV	-0.1809	-0.1695	0.9452	0.9636	0.9957	0.9945	0.9971	0.9982	1.0000						
modular_opndV	-0.1592	-0.1481	0.9119	0.9450	0.9871	0.9927	0.9980	0.9975	0.9914	1.0000					
V	-0.1319	-0.1203	0.9739	0.9879	0.9978	0.9943	0.9883	0.9871	0.9902	0.9790	1.0000				
sumV	-0.1299	-0.1183	0.9767	0.9890	0.9970	0.9929	0.9860	0.9849	0.9886	0.9760	0.9999	1.0000			
optrV	-0.1304	-0.1188	0.9823	0.9907	0.9948	0.9885	0.9809	0.9798	0.9854	0.9686	0.9988	0.9993	1.0000		
opndV	-0.1288	-0.1172	0.9666	0.9844	0.9978	0.9968	0.9911	0.9898	0.9908	0.9842	0.9992	0.9987	0.9961	1.0000	
LOC	-0.1246	-0.1122	0.9322	0.9502	0.9738	0.9720	0.9685	0.9647	0.9628	0.9625	0.9702	0.9691	0.9665	0.9706	1.0000

Correlations between dynamic and static measures obtained by using the MY_VOL and CSIZE tools for all programs in the test suite

	st-size1	st-size2	modular_n	modularN	V	LOC
st-size1	1.0000					
st-size2	0.9981	1.0000				
modular_n	-0.0884	-0.0758	1.0000			
modularN	-0.1355	-0.1231	0.9608	1.0000		
V	-0.1115	-0.0992	0.9700	0.9973	1.0000	
LOC	-0.1246	-0.1122	0.9745	0.9886	0.9903	1.0000

APPENDIX D

DYNAMIC AND STATIC MEASURES FOR INDIVIDUAL APPLICATION AREAS

All of the dynamic and static measures in this appendix are as defined in Subsection 4.3

DYNAMIC MEASURES FOR THE STRING PROCESSING PROGRAMS ORDERED BY st-size1

PROGRAM	st-size1	st-size2
fill	781	441
expand	784	441
par	875	487
uuconvert	908	504
rs	983	551
agrep	1102	638
include	1115	627
friends	1167	652
opcom	1230	690
indent	1356	887
gzip	1609	1055
ce	1611	929
forwarder	1925	1101
fplan	2009	1163
archie	2285	1260
man	2388	1509
utree	2506	1436
combine	4691	2530
convert	4691	2530
import	4691	2530
mogrify	4691	2530
montage	4707	2538
segment	4753	2569
sail	4826	2705
sc	4850	2778
display	5087	2729
xnlock	7768	4535
ghostview	9208	5036
xgopher	10305	5634

STATIC MEASURES OBTAINED BY USING THE HALSTEAD TOOL FOR THE STRING PROCESSING PROGRAMS

PROGRAM	modular_n_1 / modular_n_2	modularN_1 / modularN_2	modularV / modular_sumV	modular_optrV / modular_opndV	V / sumV	optrV / opndV
agrep	445	1109	13339	9780	170454	138946	71051	67895	245102	216277	117352	98925
archie	536	1490	7926	5513	96447	78179	40676	37506	147620	129971	71858	58113
ce	246	1080	11551	7971	158220	125876	63472	62404	202499	172066	91744	80322
combine	134	634	2912	1709	34574	26662	14370	12290	44292	36484	20576	15908
convert	138	668	3602	2087	43399	33610	18276	15333	54925	45189	25605	19584
display	161	1219	16045	11621	258312	198191	90750	107440	288569	236757	117625	119132
expand	40	40	254	154	2579	2171	1352	820	2579	2171	1352	820
fill	45	103	955	647	11550	9571	5245	4326	11550	9571	5245	4326
forwarder	46	106	493	356	5368	4277	2206	2072	6153	5118	2723	2395
fplan	360	1029	7274	5632	95287	77478	37933	39547	134736	118129	61770	56360
friends	32	40	145	91	1456	1209	725	484	1456	1209	725	484
ghostview	695	3674	26676	20796	394285	314643	146466	168178	574083	498134	251845	246290
gzip	688	2522	15352	12517	223297	182405	86330	96075	324628	286159	144712	141447
import	132	645	3186	1876	38029	29263	15687	13576	48604	39952	22443	17509

STATIC MEASURES OBTAINED BY USING THE HALSTEAD TOOL FOR THE STRING PROCESSING PROGRAMS (cont.)

PROGRAM	modular_n_1 / modular_n_2		modularN$_1$ / modularN$_2$		modularV / modular_sumV		modular_optrV / modular_opndV		V / sumV		optrV / opndV	
include	35	58	347	215	3568	2923	1704	1218	3675	3039	1780	1259
indent	410	1553	12155	8436	164000	132220	67834	64382	225242	194928	105499	89429
man	201	367	1630	1109	17142	13968	7717	6250	25061	21919	12471	9448
mogrify	139	675	3707	2172	45042	34948	18913	16033	56843	46804	26390	20414
montage	153	787	5657	3501	73409	57222	29848	27373	90449	74735	41055	33680
opcom	74	155	1021	649	11989	9707	5342	4364	13091	11062	6340	4722
par	216	563	5051	3769	66209	54168	27725	26444	84720	73607	39170	34437
rs	38	113	856	538	10090	8161	4492	3669	10090	8161	4492	3669
sail	624	2011	18682	14185	251910	201483	97378	104103	373487	329132	173470	155662
sc	712	2922	26551	17343	362998	289280	149995	139279	519149	451256	251590	199665
segment	152	725	5186	3401	68273	54372	28295	26076	83950	69904	37588	32316
utree	897	3946	31440	20860	427183	337901	173270	164630	640240	557591	308394	249197
uuconvert	36	68	405	298	4710	3908	2094	1814	4710	3908	2094	1814
xgopher	1696	5661	29174	23301	394498	314025	145608	168418	674036	603466	312976	290490
xnlock	70	398	2528	1918	38101	30271	14243	16028	39438	32060	15495	16565

STATIC MEASURES OBTAINED BY USING THE METRE TOOL FOR THE STRING PROCESSING PROGRAMS

PROGRAM	modular_n_1 / modular_n_2		modularN_1 / modularN_2		modularV / modular_sumV		modular_optrV / modular_opndV		V / sumV		optrV / opndV	
agrep	1160	1763	12620	8877	131499	106568	57983	48592	247500	224198	128470	95728
archie	542	972	5438	3631	56977	45436	24658	20779	95806	85426	49389	36037
ce	1201	1715	10945	7231	122498	95900	50883	45019	209202	189657	111968	77690
combine	42	191	1204	805	14949	11631	5857	5774	15799	12592	6492	6100
convert	44	302	1860	1222	24087	18382	9087	9295	25996	20222	10155	10067
display	543	1888	13845	10236	181500	143389	69791	73600	270847	237174	125779	111395
expand	52	42	475	210	3897	3239	2286	953	4490	3840	2708	1132
fill	91	124	1059	622	10566	8621	5141	3480	13025	11217	6892	4326
forwarder	92	144	401	293	3563	2729	1406	1323	5471	4717	2616	2101
fplan	764	1127	5044	3641	48739	38365	20389	17979	94536	85222	48309	36913
friends	22	32	81	61	817	666	361	305	817	666	361	305
ghostview	1683	2728	15187	10226	156975	123295	65782	57512	307672	279472	162756	116716
gzip	1659	2205	11952	7829	116348	94092	54202	39887	235708	214793	127840	86953
import	41	233	1469	1012	18966	14569	7049	7520	20091	15829	7870	7959

STATIC MEASURES OBTAINED BY USING THE METRE TOOL FOR THE STRING PROCESSING PROGRAMS (cont.)

PROGRAM	modular_n_1 / modular_n_2	modularN_1 / modularN_2	modularV / modular_sumV	modular_optrV / modular_opndV	V / sumV	optrV / opndV
include	67	96	263	179	2167	1662	916	746	3248	2774	1595	1179
indent	550	1178	9275	6213	110892	88139	46289	41851	166572	147819	84433	63386
man	138	291	1003	751	9719	7632	3887	3746	15338	13277	7130	6147
mogrify	47	275	1949	1267	25426	19611	9874	9737	26792	21093	10826	10267
montage	94	447	3614	2565	48298	37959	18612	19347	56102	46271	23688	22583
opcom	166	196	1003	561	8275	6658	4199	2459	13294	11669	7397	4272
par	561	754	4708	3070	47944	38841	22290	16556	80587	72337	42993	29344
rs	140	213	727	509	7265	5810	3136	2673	10461	9120	5183	3937
sail	1667	2898	13578	10191	147850	119464	62037	57431	288945	262531	145326	117205
sc	2925	4532	21991	14261	221999	174321	96273	78054	466359	426422	253209	173213
segment	277	456	3063	2180	32675	26192	14035	12160	49901	44108	24852	19256
utree	4100	6633	27953	17311	271347	216146	123720	92411	606074	555246	335475	219771
uuconvert	63	83	392	274	3631	2902	1600	1302	4788	4090	2343	1747
xgopher	3216	5532	24487	16437	239773	181182	93626	87574	535889	489670	285299	204371
xnlock	253	551	2015	1551	21666	17342	8760	8583	34416	30209	16086	14123

STATIC MEASURES OBTAINED BY USING THE MY_VOL AND CSIZE TOOLS FOR THE STRING PROCESSING PROGRAMS

PROGRAM	modular_n	modularN	modularV	V	LOC
agrep	1788	23828	175228	257441	3465
archie	2024	13818	99287	151763	2918
ce	1317	19717	159795	204328	4238
combine	760	4454	33316	42624	1067
convert	799	5534	42253	53359	1367
display	1375	27354	255843	285171	4898
expand	80	429	2712	2712	106
fill	148	1589	11456	11456	324
forwarder	153	872	5514	6328	189
fplan	1381	13095	96905	136600	2192
friends	73	237	1467	1467	51
ghostview	4357	47920	397381	579311	8475
gzip	3147	27083	216236	314698	5220
import	769	4873	36606	46717	1131
include	86	561	3605	3605	113
indent	1949	21769	173151	237903	4660
man	566	2752	17218	25166	626
mogrify	807	5715	43827	55186	1374
montage	931	8905	71510	87827	1805
opcom	230	1656	11179	12992	311
par	783	9112	68437	87592	1430
rs	152	1380	10002	1002	343
sail	2644	32715	250657	371921	4147
sc	3616	44648	369433	527749	8343
segment	874	8290	66103	81006	1749
utree	4824	51228	418341	626827	9413
uuconvert	103	681	4554	4554	121
xgopher	7355	53759	404042	690508	11769
xnlock	470	4428	37964	39305	708

DYNAMIC MEASURES FOR THE NETWORK PROGRAMS
ORDERED by st-size1

PROGRAMS	st-size1	st-size2
udp_tli_server	965	542
tcp_tli_server	989	554
udp_server	1000	560
udp_sock_server	1255	698
tcp_sock_server	1267	704
udp_tli_client	1329	736
tcp_tli_client	1341	742
tcp_client	1366	755
udp_client	1421	787
udp_sock_client	1662	916
tcp_sock_client	1676	923
conc_serv	3146	1804
rpc	3997	2224

STATIC MEASURES OBTAINED BY USING THE HALSTEAD TOOL FOR THE NETWORK PROGRAMS

PROGRAM	modular_n_1	modular_n_2	modularN_1	modularN_2	modularV	modular_sumV	modular_optrV	modular_opndV	V	sumV	optrV	opndV
conc_serv	56	117	402	243	3951	3155	1807	1348	4795	4004	2335	1669
rpc	46	56	212	128	1935	1574	956	617	2269	1914	1171	743
tcp_client	31	65	268	163	2838	2309	1328	982	2838	2309	1328	982
tcp_sock_client	21	41	134	90	1334	1071	589	482	1334	1071	589	482
tcp_sock_server	24	41	162	103	1596	1295	743	552	1596	1295	743	552
tcp_tli_client	22	53	200	125	2024	1608	892	716	2024	1608	892	716
tcp_tli_server	23	62	300	183	3096	2447	1357	1090	3096	2447	1357	1090
udp_client	32	70	309	207	3443	2814	1545	1269	3443	2814	1545	1269
udp_server	27	73	298	187	3222	2574	1417	1157	3222	2574	1417	1157
udp_sock_client	21	39	146	94	1418	1138	641	497	1418	1138	641	497
udp_sock_server	18	34	131	86	1237	984	546	438	1237	984	546	438
udp_tli_client	23	55	191	119	1948	1552	864	688	1948	1552	864	688
udp_tli_server	20	53	220	138	2216	1741	951	790	2216	1741	951	790

STATIC MEASURES OBTAINED BY USING THE METRE TOOL FOR THE NETWORK PROGRAMS

PROGRAM	modular_n_1 / modular_n_2		modularN_1 / modularN_2		modularV / modular_sumV		modular_optrV / modular_opndV		V / sumV		optrV / opndV	
conc_serv	39	105	271	197	2736	2133	1079	1054	3356	2755	1432	1323
rpc	31	58	156	111	1330	1023	541	483	1729	1423	773	650
tcp_client	20	58	182	132	1974	1560	787	773	1974	1560	787	773
tcp_sock_client	14	30	86	69	846	666	327	339	846	666	327	339
tcp_sock_server	18	40	128	91	1283	1018	534	484	1283	1018	534	484
tcp_tli_client	14	42	120	94	1243	964	457	507	1243	964	457	507
tcp_tli_server	16	58	209	154	2254	1738	836	902	2254	1738	836	902
udp_client	29	65	219	168	2403	1919	956	963	2537	2076	1064	1012
udp_server	18	59	199	146	2162	1689	830	859	2162	1689	830	859
udp_sock_client	14	30	97	75	939	737	369	368	939	737	369	368
udp_sock_server	12	32	100	76	961	738	358	380	961	738	358	380
udp_tli_client	14	44	114	89	1189	920	434	486	1189	920	434	486
udp_tli_server	13	46	140	110	1471	1126	518	608	1471	1126	518	608

STATIC MEASURES OBTAINED BY USING THE MY_VOL AND CSIZE TOOLS FOR THE NETWORK PROGRAMS

PROGRAM	modular_n	modularN	modularV	V	LOC
conc_serv	3873	172	634	4708	119
rpc	1924	100	353	2345	83
tcp_client	2851	96	433	2851	86
tcp_sock_client	1304	62	219	1304	40
tcp_sock_server	1596	65	265	1596	52
tcp_tli_client	1962	75	315	1962	53
tcp_tli_server	3038	85	474	3038	83
udp_client	3450	102	517	3450	95
udp_server	3149	100	474	3149	81
udp_sock_client	1370	60	232	1370	39
udp_sock_server	1220	52	214	1220	39
udp_tli_client	1879	78	299	1879	47
udp_tli_server	2148	73	347	2148	52

DYNAMIC MEASURES FOR THE NUMERIC PROGRAMS ORDERED BY st-size1

PROGRAM	st-size1	st-size2
uuconvert	908	504
stat	1044	582
fplan	2009	1163
lander	2262	1262
othello	2342	1303
xdl	2643	1432
xasteroids	2856	1558
combine	4691	2530
convert	4691	2530
import	4691	2530
mogrify	4691	2530
montage	4707	2538
segment	4753	2569
animate	4803	2581
sail	4826	2705
sc	4850	2778
display	5087	2729

STATIC MEASURES OBTAINED BY USING THE HALSTEAD TOOL FOR THE NUMERIC PROGRAMS

PROGRAM	modular_n_1 / modular_n_2		modularN_1 / modularN_2		modularV / modular_sumV		modular_optrV / modular_opndV		V / sumV		optrV / opndV	
animate	171	1024	8473	5399	120469	93097	46883	46214	141811	116841	62851	53990
combine	134	634	2912	1709	34574	26662	14370	12290	44292	36484	20576	15908
convert	138	668	3602	2087	43399	33610	18276	15333	54925	45189	25605	19584
display	161	1219	16045	11621	258312	198191	90750	107440	288569	236757	117625	119132
fplan	360	1029	7274	5632	95287	77478	37933	39547	134736	118129	61770	56360
import	132	645	3186	1876	38029	29263	15687	13576	48604	39952	22443	17509
lander	177	489	2157	1729	28215	23232	11450	11782	36448	31554	16108	15446
mogrify	139	675	3707	2172	45042	34948	18913	16033	56843	46804	26390	20414
montage	153	787	5657	3501	73409	57222	29848	27373	90449	74735	41055	33680
othello	47	146	2420	1774	31843	26197	13442	12755	31843	26197	13442	12755
sail	624	2011	18682	14185	251910	201483	97378	104103	373487	329132	173470	155662
sc	712	2922	26551	17343	362998	289280	149995	139279	519149	451256	251590	199665
segment	152	725	5186	3401	68273	54372	28295	26076	83950	69904	37588	32316
stat	39	104	483	339	5885	4824	2553	2271	5885	4824	2553	2271
uuconvert	36	68	405	298	4710	3908	2094	1814	4710	3908	2094	1814
xasteroids	54	361	3468	2921	55565	44774	19958	24816	55565	44774	19958	24816
xdl	47	215	1204	957	17360	14103	6688	7415	17360	14103	6688	7415

STATIC MEASURES OBTAINED BY USING THE METRE TOOL FOR THE NUMERIC PROGRAMS

PROGRAM	modular_n_1	modular_n_2	modularN$_1$	modularN$_2$	modularV	modular_sumV	modular_optrV	modular_opndV	V	sumV	optrV	opndV
animate	162	834	6167	4397	84944	66306	31818	34487	105217	87933	45265	42668
combine	42	191	1204	805	14949	11631	5857	5774	15799	12592	6492	6100
convert	44	302	1860	1222	24087	18382	9087	9295	25996	20222	10155	10067
display	543	1888	13845	10236	181500	143389	69791	73600	270847	237174	125779	111395
fplan	764	1127	5044	3641	48739	38365	20389	17979	94536	85222	48309	36913
import	41	233	1469	1012	18966	14569	7049	7520	20091	15829	7870	7959
lander	289	493	1727	1242	16474	13051	6975	6074	28535	25228	14118	11110
mogrify	47	275	1949	1267	25426	19611	9874	9737	26792	21093	10826	10267
montage	94	447	3614	2565	48298	37959	18612	19347	56102	46271	23688	22583
othello	236	358	1806	1396	17991	14452	7685	6766	29504	26079	14236	11843
sail	1667	2898	13578	10191	147850	119464	62037	57431	288945	262531	145326	117205
sc	2925	4532	21991	14261	221999	174321	96273	78054	466359	426422	253209	173213
segment	277	456	3063	2180	32675	26192	14035	12160	49901	44108	24852	19256
stat	38	107	340	250	3924	3070	1499	1570	4236	3470	1784	1685
uuconvert	63	83	392	274	3631	2902	1600	1302	4788	4090	2343	1747
xasteroids	278	598	2754	2262	31699	25239	12169	13070	49030	43224	22360	20865
xdl	86	245	1204	866	14542	11601	5850	5751	17327	14610	7737	6873

STATIC MEASURES OBTAINED BY USING THE MY_VOL AND CSIZE TOOLS FOR THE NUMERIC PROGRAMS

PROGRAM	modular_n	modularN	modularV	V	LOC
animate	116395	1186	13382	136655	2533
combine	33316	760	4454	42624	1067
convert	42253	799	5534	53359	1367
display	255843	1375	27354	285171	4898
fplan	96905	1381	13095	136600	2192
import	36606	769	4873	46717	1131
lander	28148	665	3870	36290	825
mogrify	43827	807	5715	55186	1374
montage	71510	931	8905	87827	1805
othello	30636	192	4039	30636	560
sail	250657	2644	32715	371921	4147
sc	369433	3616	44648	527749	8343
segment	66103	874	8290	81006	1749
stat	6000	141	822	5869	155
uuconvert	4554	103	681	4554	121
xasteroids	56714	426	6493	56714	792
xdl	17677	261	2202	17677	371

DYNAMIC MEASURES FOR THE GRAPHICS PROGRAMS
ORDERED BY st-size1

PROGRAM	st-size1	st-size2
form_corners	9033	4846
tictactoe	9034	5650
traversal	9069	4864
warning	9071	4869
modal	9073	4870
reason	9073	4870
dialog	9077	4872
arrow	9083	4872
hello_dialog	9085	4876
show_pix	9088	4881
ask_user_simple	9088	4878
map_dlg	9096	4882
ask_user	9190	4939
ghostview	9208	5036
xcal	9465	5082
action_area	9787	5238
spreadsheet	10249	4747
xgopher	10305	5634
paned_win1	10401	5559
paned_win2	10401	5559
unit_types	10401	5559
text_entry	10568	5476
help_text	10867	5813
prompt_dlg	10931	5842
select_dlg	11014	5890
modify_btn	11019	5887
cmd_area	11217	6000
file_sel	11312	6051
dynapix	11394	6094
msg_area	11395	6095
xgetftp	12086	6480

PROGRAM	st-size1	st-size2
othello	2342	1303
xdl	2643	1432
xasteroids	2856	1558
combine	4691	2530
import	4691	2530
animate	4803	2581
display	5087	2729
xnlock	7768	4535
xtalk	7844	4222
xcalendar	7915	4262
xpostit	7946	4271
xalarm	8170	4384
radiobox	8704	4666
entry_cb	8718	4673
pixmaps	8738	4683
toggle	8738	4683
frame	8747	4690
toggle_box	8777	4707
drawing	8779	4707
draw2	8780	4708
arrow_timer	8782	4708
rowcol	8798	4717
hello	8800	4718
proc_traversal	8800	4718
pushb	8800	4718
multi_click	8815	4726
simple_radio	8824	4730
corners	8960	4806
drawn	9025	4841
attach	9032	4845

STATIC MEASURES OBTAINED BY USING THE HALSTEAD TOOL FOR THE GRAPHICS PROGRAMS

PROGRAM	modular_n_1 / modular_n_2		modularN_1 / modularN_2		modularV / modular_sumV		modular_optrV / modular_opndV		V / sumV		optrV / opndV	
action_area	32	119	416	395	5870	4803	2080	2723	5870	4803	2080	2723
animate	171	1024	8473	5399	120469	93097	46883	46214	141811	116841	62851	53990
arrow	16	40	133	111	1417	1123	532	591	1417	1123	532	591
arrow_timer	27	73	210	196	2697	2212	999	1213	2697	2212	999	1213
ask_user	28	92	303	264	3916	3179	1457	1722	3916	3179	1457	1722
ask_user_simple	18	40	102	80	1066	851	425	426	1066	851	425	426
attach	15	30	82	72	846	674	320	353	846	674	320	353
cmd_area	25	89	252	210	3157	2530	1170	1360	3157	2530	1170	1360
combine	134	634	2912	1709	34574	26662	14370	12290	44292	36484	20576	15908
corners	21	61	191	163	2251	1806	839	967	2251	1806	839	967
dialog	17	57	158	153	1931	1538	646	892	1931	1538	646	892
display	161	1219	16045	11621	258312	198191	90750	107440	288569	236757	117625	119132
draw2	22	80	285	264	3663	2940	1271	1669	3663	2940	1271	1669
drawing	20	72	174	165	2211	1770	752	1018	2211	1770	752	1018
drawn	21	64	184	169	2263	1822	808	1014	2263	1822	808	1014
dynapix	26	139	487	433	6777	5372	2289	3083	6777	5372	2289	3083
entry_cb	20	42	110	97	1233	998	475	523	1233	998	475	523
file_sel	35	96	356	258	4319	3525	1826	1699	4319	3525	1826	1699
form_corners	14	29	93	74	906	714	354	359	906	714	354	359
frame	17	40	152	126	1622	1292	621	671	1622	1292	621	671
ghostview	695	3674	26676	20796	394285	314643	146466	168178	574083	498134	251845	246290
hello	15	32	56	58	633	509	219	290	633	509	219	290

STATIC MEASURES OBTAINED BY USING THE HALSTEAD TOOL FOR THE GRAPHICS PROGRAMS (cont.)

PROGRAM	modular_n_1	modular_n_2	modularN_1	modularN_2	modularV	modular_sumV	modular_optrV	modular_opndV	V	sumV	optrV	opndV
hello_dialog	17	52	131	126	1570	1254	535	718	1570	1254	535	718
help_text	28	170	607	521	8606	6778	2918	3860	8606	6778	2918	3860
import	132	645	3186	1876	38029	29263	15687	13576	48604	39952	22443	17509
map_dlg	22	63	153	145	1910	1549	682	867	1910	1549	682	867
modal	21	62	161	151	1989	1606	707	899	1989	1606	707	899
modify_btn	23	82	264	249	3444	2777	1194	1583	3444	2777	1194	1583
msg_area	25	146	510	458	7180	5661	2368	3293	7180	5661	2368	3293
multi_click	20	49	113	103	1319	1067	488	578	1319	1067	488	578
othello	47	146	2420	1774	31843	26197	13442	12755	31843	26197	13442	12755
paned_win1	14	36	80	67	830	651	305	346	830	651	305	346
paned_win2	14	46	119	106	1329	1039	453	585	1329	1039	453	585
pixmaps	29	50	129	100	1444	1191	627	564	1444	1191	627	564
proc_traversal	15	33	84	79	910	727	328	399	910	727	328	399
prompt_dlg	18	62	151	151	1909	1529	630	899	1909	1529	630	899
pushb	19	41	98	88	1099	888	416	471	1099	888	416	471
radiobox	19	42	109	98	1228	991	463	528	1228	991	463	528
reason	25	64	184	157	2208	1796	854	942	2208	1796	854	942
rowcol	15	21	62	49	574	457	242	215	574	457	242	215
select_dlg	33	95	337	284	4347	3566	1700	1866	4347	3566	1700	1866
show_pix	29	77	228	194	2839	2323	1108	1216	2839	2323	1108	1216

STATIC MEASURES OBTAINED BY USING THE HALSTEAD TOOL FOR THE GRAPHICS PROGRAMS (cont.)

PROGRAM	modular_n_1	modular_n_2	modularN_1	modularN_2	modularV	modular_sumV	modular_optrV	modular_opndV	V	sumV	optrV	opndV
simple_radio	17	37	93	87	1036	833	380	453	1036	833	380	453
spreadsheet	23	37	95	80	1034	846	430	417	1034	846	430	417
text_entry	17	39	99	74	1005	796	405	391	1005	796	405	391
tictactoe	22	58	129	122	1587	1290	575	715	1587	1290	575	715
toggle	20	57	134	123	1611	1297	579	717	1611	1297	579	717
toggle_box	25	54	186	145	2087	1698	864	834	2087	1698	864	834
traversal	23	67	153	141	1909	1547	692	855	1909	1547	692	855
unit_types	14	37	82	69	857	672	312	359	857	672	312	359
warning	18	48	104	94	1197	959	434	525	1197	959	434	525
xalarm	354	1449	8264	5563	106147	84708	43759	40946	149555	128392	69976	58416
xasteroids	54	361	3468	2921	55565	44774	19958	24816	55565	44774	19958	24816
xcal	34	103	343	267	4330	3530	1745	1785	4330	3530	1745	1785
xcalendar	221	986	5508	4557	81300	65560	30017	35543	103038	88217	42896	45321
xdl	47	215	1204	957	17360	14103	6688	7415	17360	14103	6688	7415
xgetftp	309	1326	7987	6057	109055	85710	40219	45491	149921	128893	66064	62828
xgopher	1696	5661	29174	23301	394498	314025	145608	168418	674036	603466	312976	290490
xnlock	70	398	2528	1918	38101	30271	14243	16028	39438	32060	15495	16565
xpostit	234	747	2597	2158	34577	27895	12958	14937	47256	41037	20439	20598
xtalk	30	167	791	602	10618	8326	3881	4445	10618	8326	3881	4445

STATIC MEASURES OBTAINED BY USING THE METRE TOOL FOR THE GRAPHICS PROGRAMS

PROGRAM	modular_n_1	modular_n_2	modularN_1	modularN_2	modularV	modular_sumV	modular_optrV	modular_opndV	V	sumV	optrV	opndV
action_area	59	152	282	263	2965	2266	964	1303	4208	3565	1659	1906
animate	162	834	6167	4397	84944	66306	31818	34487	105217	87933	45265	42668
arrow	6	38	113	107	1201	854	292	562	1201	854	292	562
arrow_timer	32	79	154	141	1588	1194	495	698	2004	1659	770	889
ask_user	38	92	202	181	1999	1524	660	864	2690	2241	1060	1181
ask_user_simple	12	33	78	66	791	613	280	333	791	613	280	333
attach	5	27	60	58	590	415	139	276	590	415	139	276
cmd_area	23	83	231	180	2422	1752	746	1007	2765	2192	1045	1148
combine	42	191	1204	805	14949	11631	5857	5774	15799	12592	6492	6100
corners	21	51	142	126	1378	1061	475	586	1654	1338	624	715
dialog	24	62	135	110	1180	839	356	483	1574	1274	619	655
display	543	1888	13845	10236	181500	143389	69791	73600	270847	237174	125779	111395
draw2	19	75	244	216	2555	1908	782	1124	3015	2382	1036	1345
drawing	20	63	141	125	1427	1082	465	618	1696	1357	609	747
drawn	19	69	146	133	1526	1141	459	682	1802	1433	620	812
dynapix	53	168	381	339	4281	3199	1286	1913	5607	4688	2182	2506
entry_cb	21	35	70	60	635	492	241	251	755	615	307	308
file_sel	58	94	255	186	2351	1858	991	867	3196	2713	1494	1219
form_corners	6	22	62	60	586	428	160	268	586	428	160	268
frame	5	33	83	81	861	601	193	409	861	601	193	409
ghostview	1683	2728	15187	10226	156975	123295	65782	57512	307672	279472	162756	116716
hello	7	23	36	34	318	223	81	143	343	255	101	154

STATIC MEASURES OBTAINED BY USING THE METRE TOOL FOR THE GRAPHICS PROGRAMS (cont.)

PROGRAM	modular_n_1	modular_n_2	modularN$_1$	modularN$_2$	modularV	modular_sumV	modular_optrV	modular_opndV	V	sumV	optrV	opndV
help_text	43	180	545	454	6447	4817	2014	2803	7793	6359	2957	3401
import	41	233	1469	1012	18966	14569	7049	7520	20091	15829	7870	7959
map_dlg	23	57	130	103	1115	827	386	441	1473	1189	588	601
modal	17	59	132	113	1254	921	384	537	1531	1204	540	665
modify_btn	34	101	230	197	2209	1613	672	942	3022	2482	1170	1312
msg_area	52	179	402	359	4588	3420	1355	2064	5975	4978	2292	2687
multi_click	20	45	75	67	660	479	199	279	855	692	324	368
othello	236	358	1806	1396	17991	14452	7685	6766	29504	26079	14236	11843
paned_win1	5	32	57	55	583	407	132	275	583	407	132	275
paned_win2	7	43	101	95	1106	799	284	515	1106	799	284	515
pixmaps	26	43	99	79	948	735	366	369	1087	894	465	429
proc_traversal	8	26	56	52	504	353	127	227	549	412	168	244
prompt_dlg	18	62	127	112	1167	837	332	506	1511	1196	530	667
pushb	14	36	71	65	636	463	188	275	768	606	270	336
radiobox	15	41	78	74	743	536	202	334	883	701	305	396
reason	29	72	146	122	1365	1028	462	565	1784	1462	709	753
rowcol	5	17	40	35	334	236	93	143	334	236	93	143
select_dlg	40	106	225	199	2221	1698	749	950	3048	2536	1197	1339
show_pix	37	76	153	125	1416	1077	498	578	1896	1578	797	781

STATIC MEASURES OBTAINED BY USING THE METRE TOOL FOR THE GRAPHICS PROGRAMS (cont.)

PROGRAM	modular_n_1	modular_n_2	modularN_1	modularN_2	modularV	modular_sumV	modular_optrV	modular_opndV	V	sumV	optrV	opndV
spreadsheet	14	28	66	57	663	525	251	274	663	525	251	274
text_entry	12	30	75	58	717	553	269	285	717	553	269	285
tictactoe	22	50	99	89	987	760	337	422	1160	944	441	502
toggle	17	51	104	98	1108	831	328	503	1230	981	425	556
toggle_box	36	53	128	102	1186	921	459	461	1489	1246	662	584
traversal	24	58	122	107	1229	953	432	522	1456	1186	559	627
unit_types	5	33	59	57	609	425	137	288	609	425	137	288
warning	12	43	72	67	671	489	189	299	804	622	258	364
xalarm	914	1624	7061	4647	68245	54188	30052	24132	132411	119014	69452	49562
xasteroids	278	598	2754	2262	31699	25239	12169	13070	49030	43224	22360	20865
xcal	51	104	237	198	2483	1922	891	1032	3165	2671	1344	1327
xcalendar	403	773	2838	2154	28963	22431	11122	11310	50917	45228	24562	20666
xdl	86	245	1204	866	14542	11601	5850	5751	17327	14610	7737	6873
xgetftp	954	1988	7628	5421	74452	53933	25337	28593	150358	134899	75501	59398
xgopher	3216	5532	24487	16437	239773	181182	93626	87574	535889	489670	285299	204371
xnlock	253	551	2015	1551	21666	17342	8760	8583	34416	30209	16086	14123
xpostit	286	542	2481	1601	23454	17710	9362	8349	39569	34785	20245	14541
xtalk	44	157	864	605	10295	7786	3710	4076	11239	9130	4717	4413

STATIC MEASURES OBTAINED BY USING THE MY_VOL AND CSIZE TOOLS FOR THE GRAPHICS PROGRAMS

PROGRAM	modular_n	modularN	modularV	V	LOC
action_area	151	837	6059	6059	168
animate	1186	13382	116395	136655	2533
arrow	56	248	1440	1440	59
arrow_timer	101	423	2816	2816	82
ask_user	121	592	4096	4096	121
ask_user_simple	107	442	2980	2980	101
attach	45	157	862	862	34
cmd_area	114	464	3170	3170	78
combine	760	4454	33316	42624	1067
corners	82	361	2295	2295	76
dialog	74	323	2006	2006	66
display	1375	27354	255843	285171	4898
draw2	102	546	3643	3643	83
drawing	92	347	2264	2264	59
drawn	84	356	2276	2276	65
dynapix	164	917	6747	6747	174
entry_cb	62	218	1298	1298	46
file_sel	132	613	4318	4318	119
form_corners	43	166	901	901	37
frame	57	281	1639	1639	77
ghostview	4357	47920	397381	579311	8475
hello	47	121	672	672	29
hello_dialog	69	268	1637	1637	53
help_text	198	1137	8675	8675	237
import	769	4873	36606	46717	1131
map_dlg	85	306	1961	1961	61
modal	83	319	2034	2034	64
modify_btn	105	524	3518	3518	100
msg_area	170	966	7157	7157	185

STATIC MEASURES OBTAINED BY USING THE MY_VOL AND CSIZE TOOLS FOR THE GRAPHICS PROGRAMS (cont.)

PROGRAM	modular_n	modularN	modularV	V	LOC
othello	192	4039	30636	30636	560
paned_win1	50	149	841	841	32
paned_win2	60	230	1359	1359	44
pixmaps	78	234	1471	1471	49
proc_traversal	48	167	933	933	34
prompt_dlg	80	310	1960	1960	57
pushb	60	188	1110	1110	38
radiobox	62	215	1280	1280	40
reason	90	358	2324	2324	65
rowcol	36	111	574	574	21
select_dlg	128	646	4522	4522	128
show_pix	58	178	1043	1043	37
simple_radio	55	187	1081	1081	38
spreadsheet	60	181	1069	1069	34
text_entry	56	177	1028	1028	37
tictactoe	80	261	1650	1650	53
toggle	78	263	1653	1653	54
toggle_box	80	337	2130	2130	69
traversal	90	305	1980	1980	64
unit_types	51	153	868	868	33
warning	66	204	1233	1233	43
xalarm	1793	13000	99568	140506	1869
xasteroids	426	6493	56714	56714	792
xcal	136	619	4387	4387	122
xcalendar	1211	10147	81858	103925	1722
xdl	261	2202	17677	17677	371
xgetftp	1632	13716	106283	146383	2653
xgopher	7355	53759	404042	690508	11769
xnlock	470	4428	37964	39305	708
xpostit	985	4797	34883	47701	973
xtalk	196	1378	10493	10493	230

DYNAMIC MEASURES FOR THE MOTIF PROGRAMS
ORDERED BY st-size1

PROGRAM	st-size1	st-size2
radiobox	8704	4666
entry_cb	8718	4673
pixmaps	8738	4683
toggle	8738	4683
frame	8747	4690
toggle_box	8777	4707
drawing	8779	4707
draw2	8780	4708
arrow_timer	8782	4708
rowcol	8798	4717
hello	8800	4718
proc_traversal	8800	4718
pushb	8800	4718
multi_click	8815	4726
simple_radio	8824	4730
corners	8960	4806
drawn	9025	4841
attach	9032	4845
form_corners	9033	4846
tictactoe	9034	5650
traversal	9069	4864
warning	9071	4869
modal	9073	4870

PROGRAM	st-size1	st-size2
reason	9073	4870
dialog	9077	4872
arrow	9083	4872
hello_dialog	9085	4876
show_pix	9088	4881
ask_user_simple	9088	4878
map_dlg	9096	4882
ask_user	9190	4939
xcal	9465	5082
action_area	9787	5238
spreadsheet	10249	4747
paned_win1	10401	5559
paned_win2	10401	5559
unit_types	10401	5559
text_entry	10568	5476
help_text	10867	5813
prompt_dlg	10931	5842
select_dlg	11014	5890
modify_btn	11019	5887
cmd_area	11217	6000
file_sel	11312	6051
dynapix	11394	6094
msg_area	11395	6095

STATIC MEASURES OBTAINED BY USING THE HALSTEAD TOOL FOR THE MOTIF PROGRAMS

PROGRAM	modular_n_1 / modular_n_2		modularN_1 / modularN_2		modularV / modular_sumV		modular_optrV / modular_opndV		V / sumV		optrV / opndV	
action_area	32	119	416	395	5870	4803	2080	2723	5870	4803	2080	2723
arrow	16	40	133	111	1417	1123	532	591	1417	1123	532	591
arrow_timer	27	73	210	196	2697	2212	999	1213	2697	2212	999	1213
ask_user	28	92	303	264	3916	3179	1457	1722	3916	3179	1457	1722
ask_user_simple	18	40	102	80	1066	851	425	426	1066	851	425	426
attach	15	30	82	72	846	674	320	353	846	674	320	353
cmd_area	25	89	252	210	3157	2530	1170	1360	3157	2530	1170	1360
corners	21	61	191	163	2251	1806	839	967	2251	1806	839	967
dialog	17	57	158	153	1931	1538	646	892	1931	1538	646	892
draw2	22	80	285	264	3663	2940	1271	1669	3663	2940	1271	1669
drawing	20	72	174	165	2211	1770	752	1018	2211	1770	752	1018
drawn	21	64	184	169	2263	1822	808	1014	2263	1822	808	1014
dynapix	26	139	487	433	6777	5372	2289	3083	6777	5372	2289	3083
entry_cb	20	42	110	97	1233	998	475	523	1233	998	475	523
file_sel	35	96	356	258	4319	3525	1826	1699	4319	3525	1826	1699
form_corners	14	29	93	74	906	714	354	359	906	714	354	359
frame	17	40	152	126	1622	1292	621	671	1622	1292	621	671
hello	15	32	56	58	633	509	219	290	633	509	219	290
hello_dialog	17	52	131	126	1570	1254	535	718	1570	1254	535	718
help_text	28	170	607	521	8606	6778	2918	3860	8606	6778	2918	3860
map_dlg	22	63	153	145	1910	1549	682	867	1910	1549	682	867
modal	21	62	161	151	1989	1606	707	899	1989	1606	707	899

STATIC MEASURES OBTAINED BY USING THE HALSTEAD TOOL FOR THE MOTIF PROGRAMS (cont.)

PROGRAM	modular_n_1 / modular_n_2		modularN_1 / modularN_2		modularV / modular_sumV		modular_optrV / modular_opndV		V / sumV		optrV / opndV	
msg_area	25	146	510	458	7180	5661	2368	3293	7180	5661	2368	3293
multi_click	20	49	113	103	1319	1067	488	578	1319	1067	488	578
paned_win1	14	36	80	67	830	651	305	346	830	651	305	346
paned_win2	14	46	119	106	1329	1039	453	585	1329	1039	453	585
pixmaps	29	50	129	100	1444	1191	627	564	1444	1191	627	564
proc_traversal	15	33	84	79	910	727	328	399	910	727	328	399
prompt_dlg	18	62	151	151	1909	1529	630	899	1909	1529	630	899
pushb	19	41	98	88	1099	888	416	471	1099	888	416	471
radiobox	19	42	109	98	1228	991	463	528	1228	991	463	528
reason	25	64	184	157	2208	1796	854	942	2208	1796	854	942
rowcol	15	21	62	49	574	457	242	215	574	457	242	215
select_dlg	33	95	337	284	4347	3566	1700	1866	4347	3566	1700	1866
show_pix	29	77	228	194	2839	2323	1108	1216	2839	2323	1108	1216
simple_radio	17	37	93	87	1036	833	380	453	1036	833	380	453
spreadsheet	23	37	95	80	1034	846	430	417	1034	846	430	417
text_entry	17	39	99	74	1005	796	405	391	1005	796	405	391
tictactoe	22	58	129	122	1587	1290	575	715	1587	1290	575	715
toggle	20	57	134	123	1611	1297	579	717	1611	1297	579	717
toggle_box	25	54	186	145	2087	1698	864	834	2087	1698	864	834
traversal	23	67	153	141	1909	1547	692	855	1909	1547	692	855
unit_types	14	37	82	69	857	672	312	359	857	672	312	359
warning	18	48	104	94	1197	959	434	525	1197	959	434	525
xcal	34	103	343	267	4330	3530	1745	1785	4330	3530	1745	1785

STATIC MEASURES OBTAINED BY USING THE METRE TOOL FOR THE MOTIF PROGRAMS

PROGRAM	modular_n_1	modular_n_2	modularN_1	modularN_2	modularV	modular_sumV	modular_optrV	modular_opndV	V	sumV	optrV	opndV
action_area	59	152	282	263	2965	2266	964	1303	4208	3565	1659	1906
arrow	6	38	113	107	1201	854	292	562	1201	854	292	562
arrow_timer	32	79	154	141	1588	1194	495	698	2004	1659	770	889
ask_user	38	92	202	181	1999	1524	660	864	2690	2241	1060	1181
ask_user_simple	12	33	78	66	791	613	280	333	791	613	280	333
attach	5	27	60	58	590	415	139	276	590	415	139	276
cmd_area	23	83	231	180	2422	1752	746	1007	2765	2192	1045	1148
corners	21	51	142	126	1378	1061	475	586	1654	1338	624	715
dialog	24	62	135	110	1180	839	356	483	1574	1274	619	655
draw2	19	75	244	216	2555	1908	782	1124	3015	2382	1036	1345
drawing	20	63	141	125	1427	1082	465	618	1696	1357	609	747
drawn	19	69	146	133	1526	1141	459	682	1802	1433	620	812
dynapix	53	168	381	339	4281	3199	1286	1913	5607	4688	2182	2506
entry_cb	21	35	70	60	635	492	241	251	755	615	307	308
file_sel	58	94	255	186	2351	1858	991	867	3196	2713	1494	1219
form_corners	6	22	62	60	586	428	160	268	586	428	160	268
frame	5	33	83	81	861	601	193	409	861	601	193	409
hello	7	23	36	34	318	223	81	143	343	255	101	154
hello_dialog	15	49	92	82	841	608	241	367	1044	820	359	460
help_text	43	180	545	454	6447	4817	2014	2803	7793	6359	2957	3401
map_dlg	23	57	130	103	1115	827	386	441	1473	1189	588	601
modal	17	59	132	113	1254	921	384	537	1531	1204	540	665

STATIC MEASURES OBTAINED BY USING THE METRE TOOL FOR THE MOTIF PROGRAMS (cont.)

PROGRAM	modular_n_1	modular_n_2	modularN_1	modularN_2	modularV	modular_sumV	modular_optrV	modular_opndV	V	sumV	optrV	opndV
msg_area	52	179	402	359	4588	3420	1355	2064	5975	4978	2292	2687
multi_click	20	45	75	67	660	479	199	279	855	692	324	368
paned_win1	5	32	57	55	583	407	132	275	583	407	132	275
paned_win2	7	43	101	95	1106	799	284	515	1106	799	284	515
pixmaps	26	43	99	79	948	735	366	369	1087	894	465	429
proc_traversal	8	26	56	52	504	353	127	227	549	412	168	244
prompt_dlg	18	62	127	112	1167	837	332	506	1511	1196	530	667
pushb	14	36	71	65	636	463	188	275	768	606	270	336
radiobox	15	41	78	74	743	536	202	334	883	701	305	396
reason	29	72	146	122	1365	1028	462	565	1784	1462	709	753
rowcol	5	17	40	35	334	236	93	143	334	236	93	143
select_dlg	40	106	225	199	2221	1698	749	950	3048	2536	1197	1339
show_pix	37	76	153	125	1416	1077	498	578	1896	1578	797	781
simple_radio	13	35	68	66	631	453	167	287	748	590	252	339
spreadsheet	14	28	66	57	663	525	251	274	663	525	251	274
text_entry	12	30	75	58	717	553	269	285	717	553	269	285
tictactoe	22	50	99	89	987	760	337	422	1160	944	441	502
toggle	17	51	104	98	1108	831	328	503	1230	981	425	556
toggle_box	36	53	128	102	1186	921	459	461	1489	1246	662	584
traversal	24	58	122	107	1229	953	432	522	1456	1186	559	627
unit_types	5	33	59	57	609	425	137	288	609	425	137	288
warning	12	43	72	67	671	489	189	299	804	622	258	364
xcal	51	104	237	198	2483	1922	891	1032	3165	2671	1344	1327

STATIC MEASURES OBTAINED BY USING THE MY_VOL AND CSIZE TOOLS FOR THE MOTIF PROGRAMS

PROGRAM	modular_n	modularN	modularV	V	LOC
action_area	151	837	6059	6059	168
arrow	56	248	1440	1440	59
arrow_timer	101	423	2816	2816	82
ask_user	121	592	4096	4096	121
ask_user_simple	107	442	2980	2980	101
attach	45	157	862	862	34
cmd_area	114	464	3170	3170	78
corners	82	361	2295	2295	76
dialog	74	323	2006	2006	66
draw2	102	546	3643	3643	83
drawing	92	347	2264	2264	59
drawn	84	356	2276	2276	65
dynapix	164	917	6747	6747	174
entry_cb	62	218	1298	1298	46
file_sel	132	613	4318	4318	119
form_corners	43	166	901	901	37
frame	57	281	1639	1639	77
hello	47	121	672	672	29
hello_dialog	69	268	1637	1637	53
help_text	198	1137	8675	8675	237
map_dlg	85	306	1961	1961	61
modal	83	319	2034	2034	64
modify_btn	105	524	3518	3518	100
msg_area	170	966	7157	7157	185
multi_click	69	224	1368	1368	53
paned_win1	50	149	841	841	32
paned_win2	60	230	1359	1359	44
pixmaps	78	234	1471	1471	49
proc_traversal	48	167	933	933	34

STATIC MEASURES OBTAINED BY USING THE MY_VOL AND CSIZE TOOLS FOR THE MOTIF PROGRAMS (cont.)

PROGRAM	modular_n	modularN	modularV	V	LOC
pushb	60	188	1110	1110	38
radiobox	62	215	1280	1280	40
reason	90	358	2324	2324	65
rowcol	36	111	574	574	21
select_dlg	128	646	4522	4522	128
show_pix	58	178	1043	1043	37
simple_radio	55	187	1081	1081	38
spreadsheet	60	181	1069	1069	34
text_entry	56	177	1028	1028	37
tictactoe	80	261	1650	1650	53
toggle	78	263	1653	1653	54
toggle_box	80	337	2130	2130	69
traversal	90	305	1980	1980	64
unit_types	51	153	868	868	33
warning	66	204	1233	1233	43
xcal	136	619	4387	4387	122

APPENDIX E

LEX SPECIFICATION AND SHELL SCRIPT USED IN THE HALSTEAD TOOL

LEX SPECIFICATION

```
#  This routine comes with the "Cmetrics" package taken from
#  archive of USENET newsgroup "comp.software-eng" available at
#  ftp.qucis.quensu.ca/pub/software-eng/Software/Cmetrics as
#  metrics.tar.gz.

#  This lex routine breaks up C programs into operators and operands;
#  operands go to stdout, operators go to stderr.
#  This routine expects comments and strings to be stripped first.
#  They were stripped by using "stripstr" and "stripcom" tools that come
#  along with the "Cmetrics" package.

%%
(\'.\'|\'\\.\'|\'\\[0-9]+\') {
                    /* character constants go to operand output */
                    printf(" %s ", yytext);
                    }

(\->|\+\+|\--|\<\<|\>\>|\<\=|\>\=|\=\=|\!\=|\&\&|\|\||[+*/%\-]\=)
      {
                    /* print 2 character symbols to operator output */
                    fprintf(stderr,"%s\n", yytext);
                    putchar(' ');
                    }

[\.\(\)\[\]\!\~\&\+\/\%\<\>\&\^\|\?\=\,\-] {
                    /* print most 1 char symbols to operator output */
                    fprintf(stderr,"%s\n", yytext);
                    putchar(' ');
                    }

\*/[^ \t\n0-9]      {
                    /* print indirection operator to operator output */
                    fprintf(stderr,"*indirection\n");
                    putchar(' ');
                    }
```

```
\*                  {
                    /* multiplication operator */
                    fprintf(stderr,"%s\n", yytext);
                    putchar(' ');
                    }

[\:\;\{\}]          {
                    /* delete delimiters */
                    /* NOTE - prints to stdout, unlike above */
                    putchar(' ');
                    }

sizeof/([ \t\(]|$){
                    /* sizeof operator to operator output */
                    fprintf(stderr, "sizeof\n" );
                    putchar(' ');
                    }

^#[a-z]+/[ \t\n]    {
                    /* preprocessor operators to operator output */
                    fprintf(stderr, "%s\n", yytext);
                    putchar(' ');
                    }

(if|else|while|do|for|switch|case|default|break|continue|return|goto|cas
e)/[:; \t\(\n]      {
                    /* statements */
                    fprintf(stderr, "%s\n", yytext);
                    putchar(' ');
                    }

(auto|static|extern|register|typedef)/[ \t\(\n] {
                    /* storage class specifiers */
                    fprintf(stderr, "%s\n", yytext);
                    putchar(' ');
                    }

(char|short|int|long|unsigned|float|double)/[ \t\(\)\n] {
                    /* type specifier */
                    fprintf(stderr, "%s\n", yytext);
                    putchar(' ');
                    }

(struct|union)/[ \t\(\n]       {
                    /* struct or union specifier */
                    fprintf(stderr, "%s\n", yytext);
                    putchar(' ');
                    }

[a-zA-Z_][a-zA-Z0-9_]*  {
                    /* catch operand names */
                    printf(" %s ", yytext);
                    }

.                   ECHO;
%%
```

SHELL SCRIPT TO COMPUTE THE VOLUME METRIC

```
#                                   HALSTEAD

#  This shell routine computes Halstead's Software Science Metrics
#  including the volume metric.
#  It comes with the "Cmetrics" package available from archive of USENET
#  newsgroup comp.sofware-eng at ftp site ftp.qucis.queensu.ca in the
#  directory "/pub/software-eng/software/Cmetrics.

#  A few modifications were made to the original script available with
#  the "Cmetrics" package.  Changes were made to compute vol1, VOL2,
#  vol2, n1, n2, N1, N2.


# n1 == number of unique operators
# n2 == number of unique operands
# N1 == number of total operators
# N2 == number of total operands

# open temporary files in '/tmp' directory for n1, n2, N1, N2
N1=/tmp/$$.N1
N2=/tmp/$$.N2
n1=/tmp/$$.n1
n2=/tmp/$$.n2

# open temporary file in '/tmp' directory for input and output
vol=/tmp/$$.vol
<>${vol}

# trap on interrupt
trap '/bin/rm -f ${n1} ${n2} ${N1} ${N2} ${vol} ; exit 1' 1 2 15

# if file name is not specified exit with message
if [ $# -lt 1 ]
then
  echo "usage: $0 <file> [<file>]"
  exit 1
fi

# for each file get the statistics to temp file 'vol'
for file in $*
do
  # strip comments with 'stripcom' and then pipe output to
  # 'stripstr' to strip strings.  This output is fed to
  # 'c_halsfilt' to parse it to N1
  # 'c_halsfilt' prints N1 to stderror.
  # sort the output on stdout to N2
  stripcom ${file} | stripstr |\
  c_halsfilt 2> ${N1} |\
  awk ' { for ( I = 1; I <= NF; I++) print $I; } ' |\
  sort > ${N2}


  # sort for unique terms to n1 and n2
```

```
  sort -u ${N1} > ${n1}
  uniq < ${N2} > ${n2}

  # count the no. Of lines in each files, n1, n2, N1, N2 to
  # get no. Of operators, operands, unique operators, and unique
  # operands.
  wc -l ${n1} ${n2} ${N1} ${N2} |\
  awk '
  BEGIN {
    File = "'"${file}"'";
    if ( !getline )
      printf("%s: no line count for n1\n", "'"$0"'");
    u_opators = $1;

    if ( !getline )
      printf("%s: no line count for n2\n", "'"$0"'");
    u_opands = $1;

    if ( !getline )
      printf("%s: no line count for N1\n", "'"$0"'");
    t_opators = $1;

    if ( !getline )
      printf("%s: no line count for N2\n", "'"$0"'");
    t_opands = $1;

    # program length: N = N1 + N2
    #
    pg_length = t_opators + t_opands;

    # program volume: V = N log<base2> (n1 + n2)
    #
    # CHECK TO SEE NUM. NOT EQUAL TO ZERO.
    if ((u_opators + u_opands) != 0)
    pg_volume = pg_length * (log(u_opators + u_opands)/log(2));

    opator_volume_prime = (t_opators) * (log(u_opators)/log(2));
    opand_volume_prime = (t_opands) * (log(u_opands)/log(2));

    pg_volume_prime = (opator_volume_prime) + (opand_volume_prime);


    printf("%s\t%d\t%.0f\n,\
    File, pg_length, pg_volume);
    printf("%6.0f\t%6.0f\t%6.0f\t", u_opators, u_opands, t_opators);
    printf("%6.0f\t%6.0f\t%6.0f\t", t_opands, pg_volume,
      pg_volume_prime);
    printf("%6.0f\t%6.0f\n", opator_volume_prime, opand_volume_prime);

    exit;
  }
  ' >>${vol}
done

awk '
BEGIN {
```

```
    printf("%12s\t%6s\t%6s\t%6s\t%6s\t", "module", "n1", "n2", "N1",
"N2");
    printf("%6s\t%6s\t%6s\t%6s\n", "VOL1", "vol1", "orv1", "odv1");
    exit;
}
'

cat ${vol}
cat ${vol} |\
awk '
BEGIN {
  u_opators_t = 0;
  u_opands_t = 0;
  t_opators_t = 0;
  t_opands_t = 0;
  vol1_t = 0;
  vol1_prime_t = 0;
  opr_vol1_prime_t = 0;
  opd_vol1_prime_t = 0;
  while (getline)
  {
    u_opators_t = u_opators_t + $2;
    u_opands_t = u_opands_t + $3;
    t_opators_t = t_opators_t + $4;
    t_opands_t = t_opands_t + $5;
    vol1_t += $6;
    vol1_prime_t = vol1_prime_t + $7;
    opr_vol1_prime_t += $8;
    opd_vol1_prime_t += $9;

  }

 # vol2 = (N1 + N2) * log (n1 + n2)
  vol2 = (t_opators_t + t_opands_t) * (log(u_opators_t +
   u_opands_t)/log(2));

  opr_vol2_prime = (t_opators_t) * (log(u_opators_t)/log(2));
  opd_vol2_prime = (t_opands_t) * (log(u_opands_t)/log(2));
  vol2_prime = (opr_vol2_prime) + (opd_vol2_prime);

  printf("\n");
  printf("%6s\t%6s\t%6s\t%6s\t", "n1_t", "n2_t", "N1_t", "N2_t");
  printf("%6s\t%6s\t%6s\t%6s\n", "VOL1_t", "vol1_t", "orv1_t",
"odv1_t");

  printf("%6.0f,%6.0f,%6.0f,",u_opators_t,u_opands_t,t_opators_t);
  printf("%6.0f,%6.0f,%6.0f", t_opands_t,vol1_t, vol1_prime_t);
  printf("%6.0f,%6.0f,", opr_vol1_prime_t, opd_vol1_prime_t);

  printf("%6s\t%6s\t%6s\t%6s\n", "VOL2", "vol2", "or_v2", "od_v2");
  printf("%6.0f,%6.0f,%6.0f,",vol2,vol2_prime,opr_vol2_prime);
  printf("%6.0f,", opd_vol2_prime);
}
'

/bin/rm ${n1} ${n2} ${N1} ${N2} ${vol}
exit 0
```

APPENDIX F

LEX SPECIFICATION AND SHELL SCRIPT USED IN THE MY_VOL TOOL

LEX SPECIFICATION

```
/* Lex specification to parse an input C program for MY_VOL.          */
/* This routine recognizes the tokens and prints to standard output   */
/* This was derived from the Lex specification written by Jeff Lee for */
/* April 30, 1985 ANSI C draft.  It is available at archive of USENET */
/* newsgroup comp.language.c at ftp site ftp.uu.net as                */
/* usenet.net.sources/ansi.c.grammar.Z.                               */

/* A few modifications were made to the original version to make it   */
/* suitable for the experiment                                        */

/* The tokens are printed on stdout.                                  */


D                       [0-9]
L                       [a-zA-Z_]
H                       [a-fA-F0-9]
E                       [Ee][+-]?{D}+
FS                      (f|F|l|L)
IS                      (u|U|l|L)*


%%
"/*"                    { comment(); }

"auto"                  { printf(" %s\n", yytext); }
"break"                 { printf(" %s\n", yytext); }
"case"                  { printf(" %s\n", yytext); }
"char"                  { printf(" %s\n", yytext); }
"const"                 { printf(" %s\n", yytext); }
"continue"              { printf(" %s\n", yytext); }
"default"               { printf(" %s\n", yytext); }
"do"                    { printf(" %s\n", yytext); }
"double"                { printf(" %s\n", yytext); }
"else"                  { printf(" %s\n", yytext); }
"enum"                  { printf(" %s\n", yytext); }
```

```
"extern"                { printf(" %s\n", yytext); }
"float"                 { printf(" %s\n", yytext); }
"for"                   { printf(" %s\n", yytext); }
"goto"                  { printf(" %s\n", yytext); }
"if"                    { printf(" %s\n", yytext); }
"int"                   { printf(" %s\n", yytext); }
"long"                  { printf(" %s\n", yytext); }
"register"              { printf(" %s\n", yytext); }
"return"                { printf(" %s\n", yytext); }
"short"                 { printf(" %s\n", yytext); }
"signed"                { printf(" %s\n", yytext); }
"sizeof"                { printf(" %s\n", yytext); }
"static"                { printf(" %s\n", yytext); }
"struct"                { printf(" %s\n", yytext); }
"switch"                { printf(" %s\n", yytext); }
"typedef"               { printf(" %s\n", yytext); }
"union"                 { printf(" %s\n", yytext); }
"unsigned"              { printf(" %s\n", yytext); }
"void"                  { printf(" %s\n", yytext); }
"volatile"              { printf(" %s\n", yytext); }
"while"                 { printf(" %s\n", yytext); }

{L}({L}|{D})*           { printf(" %s\n", yytext); }

0[xX]{H}+{IS}?          { printf(" %s\n", yytext); }
0{D}+{IS}?              { printf(" %s\n", yytext); }
{D}+{IS}?               { printf(" %s\n", yytext); }
'(\\.|[^\\'])+'         { printf(" %s\n", yytext); }

{D}+{E}{FS}?            { printf(" %s\n", yytext); }
{D}*"."{D}+({E})?{FS}?  { printf(" %s\n", yytext); }
{D}+"."{D}*({E})?{FS}?  { printf(" %s\n", yytext); }

\"(\\.|[^\\"])*\"       { printf(" %s\n", yytext); }

"..."                   { printf(" %s\n", yytext); }
">>="                   { printf(" %s\n", yytext); }
"<<="                   { printf(" %s\n", yytext); }
"+="                    { printf(" %s\n", yytext); }
"-="                    { printf(" %s\n", yytext); }
"*="                    { printf(" %s\n", yytext); }
"/="                    { printf(" %s\n", yytext); }
"%="                    { printf(" %s\n", yytext); }
"&="                    { printf(" %s\n", yytext); }
"^="                    { printf(" %s\n", yytext); }
"|="                    { printf(" %s\n", yytext); }
">>"                    { printf(" %s\n", yytext); }
"<<"                    { printf(" %s\n", yytext); }
"++"                    { printf(" %s\n", yytext); }
"--"                    { printf(" %s\n", yytext); }
"->"                    { printf(" %s\n", yytext); }
"&&"                    { printf(" %s\n", yytext); }
"||"                    { printf(" %s\n", yytext); }
"<="                    { printf(" %s\n", yytext); }
">="                    { printf(" %s\n", yytext); }
"=="                    { printf(" %s\n", yytext); }
"!="                    { printf(" %s\n", yytext); }
";"                     { printf(" %s\n", yytext); }
```

```
"{"                     { printf(" %s\n", yytext); }
"}"                     { /* Eat Away */}
","                     { printf(" %s\n", yytext); }
":"                     { printf(" %s\n", yytext); }
"="                     { printf(" %s\n", yytext); }
"("                     { printf(" %s\n", yytext); }
")"                     { /* Eat Away */ }
"["                     { printf(" %s\n", yytext); }
"]"                     { /* Eat Away */ }
"."                     { printf(" %s\n", yytext); }
"&"                     { printf(" %s\n", yytext); }
"!"                     { printf(" %s\n", yytext); }
"~"                     { printf(" %s\n", yytext); }
"-"                     { printf(" %s\n", yytext); }
"+"                     { printf(" %s\n", yytext); }
"*"                     { printf(" %s\n", yytext); }
"/"                     { printf(" %s\n", yytext); }
"%"                     { printf(" %s\n", yytext); }
"<"                     { printf(" %s\n", yytext); }
">"                     { printf(" %s\n", yytext); }
"^"                     { printf(" %s\n", yytext); }
"|"                     { printf(" %s\n", yytext); }
"?"                     { printf(" %s\n", yytext); }

[ \t\v\n\f]             { }
.                       { /* ignore bad characters */ }

%%


/* This function is used to ignore the comments in the program    */
/* It is called at the beginning of the comment, i,e., '/*' and   */
/* it loops until the end of the comment.                         */

comment()
{
    char c, c1;

    loop:
    while ((c = input()) != '*' && c != 0)
                putchar(c);

    if ((c1 = input()) != '/' && c != 0)
    {
      unput(c1);
      goto loop;
    }

    if (c != 0)
    putchar(c1);
}
```

SHELL SCRIPT

```
#                                   MY_VOL

#  Shell script to compute the Volume metric of given C programs.
#  This is derived from the HALSTEAD tool to compute the volume metric,
#  which comes as part of the "Cmetrics" package available at archive of
#  USENET newsgroup comp.software-eng at ftp site ftp.queensu.ca
#  in the directory /pub/software-eng/software/Cmetrics
#  It uses "stripcom" and "stripstr" available with the "Cmetrics"
#  package to remove comments and strings from input C programs.


# N = number of total tokens
# n = number of unique tokens


# temporary files N and n in the '/tmp/' directory to store
# total tokens and unique tokens, respectively.
N=/tmp/$$.N
n=/tmp/$$.n


# temporary file to store intermediate output
vol3=/tmp/$$.vol3


# open temporary file vol3 for input and output
<>${vol3}


# trap on interrupt
trap '/bin/rm -f ${n} ${N} {vol3} ; exit 1' 1 2 15


for file in $*
do
  # strip comments and strings and write output to stdout
  stripcom ${file} | stripstr |\
  filt > ${N}
  sort -u ${N} > ${n}

  wc -l ${n} ${N} |\
  awk '
  BEGIN {
    if ( !getline )
      printf ("No line count\n");
    u_tokens = $1;

    if ( !getline )
      printf ("No line count\n");
    tokens = $1;

    # prog_vol = N * log (n)/log(2);
    prog_vol = (tokens) * (log(u_tokens)/log(2));
```

```
    printf("%.0f\t%.0f\t%.0f\n", prog_vol, u_tokens, tokens);
    exit;
  }
  ' >>${vol3}

done

# for a program with more than one module compute the sum
cat ${vol3} |\
awk '
BEGIN {
  tot_vol1 = 0;
  tot_u_tokens = 0;
  tot_tokens = 0;

  while (getline)
  {
    tot_vol1 = tot_vol1 + $1;
    tot_u_tokens = tot_u_tokens + $2;
    tot_tokens = tot_tokens + $3;

  }

  tot_vol2 = (tot_tokens) * (log(tot_u_tokens)/log(2));
   printf("%.0f,%.0f,%.0f,%.0f\n",  tot_vol1,  tot_u_tokens,  tot_tokens,
tot_vol2);
}
'

/bin/rm ${n} ${N} ${vol3}
exit 0
```

APPENDIX G

ST_SIZE1 - A ROUTINE TO COMPUTE SYMBOL TABLE SIZE

```
/*                          ST_SIZE1
*
* FILE:     ST_SIZE1.c
* AUTHOR:   SRIKANTH R. MALLADI
* DATE:     08/30/96
* ADVISER:  Dr. MANSUR SAMADZADEH
*
*  This routine extracts the number of symbol table entries from object
*  file in the COFF structure given as input.
*
* USAGE:
*           ST_SIZE1 <objet file>
*/



#include <stdio.h>
#include <time.h>

#include <filehdr.h>
#include <syms.h>


FILE *fd;
FILHDR fh;              /* File header structure in COFF */

main (int argc, char *argv[])
{

/* If no. Of arguments is less than two, print error and exit     */
  if (argc < 2)
  {
    printf ("Usage: %s <object_file_name>\n", "ST_SIZE1");
    exit(1);
  }

/* Open file specified as 2nd argument for reading                */
  fd = fopen(argv[1], "r");
  if (fd == 0)
  {
```

```
    printf ("Could not open %s\n", argv[1]);
    exit(1);
  }

/* Read the file header structure in the given input COFF file    */
/* into fh.                                                       */
  fread (&fh, sizeof(FILHDR), 1, fd);

/* File header structure in a COFF file has f_nsyms member which */
/* stores the number of symbols in Symbol Table                  */
/* print the value of f_nsyms to stdout                          */
  printf ("Number of Symbols = %d\n", fh.f_nsyms);

/* close the file                                                */
  fclose(fd);
  exit(0);
}
```

APPENDIX H

ST_SIZE2 - A ROUTINE TO COMPUTE SYMBOL TABLE SIZE

```
#                                   ST_SIZE2


# FILE:      ST_SIZE2
# AUTHOR:    SRIKANTH R. MALLADI
# DATE:      08/30/96
# ADVISER:   Dr. MANSUR SAMADZADEH


#  This routine extracts the number of symbol table entries of a given
#  object file.

#  It uses the "nm" utility available on the Sequent machine running
#  DYNIX/ptx.  The "nm" utility dumps the symbol table of common object
#  file.


# temporary file in the /tmp directory to store the intermediate output
temp=/tmp/$$.temp


# trap on interrupt
trap '/bin/rm -f ${temp} ; exit 1' 1 2 15


for file in $*
do

  # dump the output of the 'nm' in temp file
  # -h flag removes header information
  # -f flag dumps full output
  # -T flag truncates the symbol name
  nm -hfT ${file} > ${temp}

  # Count no. of lines in temp file and print the output
  # which is no. Of symbols in the symbol table
  wc -l ${temp} |\
  awk '
```

```
  BEGIN {
    if ( !getline )
      printf ("No line number info.\n");
    sym_tab_size = $1;

    printf("%d\n", sym_tab_size);
    exit;
  }
  '

done

# delete the temp file
/bin/rm ${temp}
exit 0
```

VITA

Srikanth Reddy Malladi

Candidate for the Degree of

Master of Science

Thesis: INVESTIGATION OF THE RELATIONSHIP BETWEEN VOLUME METRIC AND SYMBOL TABLE SIZE

Major Field: Computer Science

Biographical:

Personal Data: Born in Warangal, Andhra Pradesh, India, August 1, 1972, son of Narasimha Reddy Malladi and Padmaja Malladi.

Education: Graduated from Government Junior College, Warangal, India, in April 1989; received Bachelor of Technology in Electrical and Electronics Engineering from Jawaharlal Nehru Technological University, Hyderabad, India, in July 1993; completed the requirements for the Master of Science Degree in Computer Science at the Computer Science Department at Oklahoma State University, in December 1996.

Professional Experience: Computer Lab Consultant, Computing and Information Services, Oklahoma State University, August 1995 to October 1996.