DEVELOPMENT AND APPLICATION OF COMPOSITE COMPLEXITY MODELS AND A RELATIVE COMPLEXITY METRIC IN A SOFTWARE MAINTENANCE ENVIRONMENT

Jonathan M. Hops
Radio Frequency and Microwave Subsystems (JPL)
California Institute of Technology, Pasadena, California, 91109

Joseph S. Sherif
Software Product Assurance (JPL)
California institute of Technology, Pasadena, California, 91109
and
California State University, Fullerton, California, 92634

ABSTRACT

A great deal of effort is now being devoted to the study, analysis, prediction and minimization of software maintenance expected cost, long before software is delivered to users or customers. It had been estimated that, on the average, the effort spent on software maintenance is as costly as the effort spent on all software costs. Software design methods should be the starting point to aid in alleviating the problems of software maintenance complexity and high costs. Two aspects of maintenance deserve attention: (1) protocols for locating defects, and rectifying them, and ensuring that no new defects are introduced in the development phase of the software process, and (2) protocols for modification, enhancement and upgrading.

This paper focuses on the second aspect - mainly, the development of protocols for helping increase the quality and reduce the costs associated with modifications, enhancements, and upgrades of existing software. This study developed parsimonious models and a relative complexity metric for complexity measurement of software that were used to rank the modules in

the system, relative to each other. Some success was achieved in using the models and relative metric to identify maintenance prone modules.

1. Introduction

A. Project Objectives

The primary objective of this study was to determine whether software metrics could help guide our efforts in the development and maintenance of the real-time embedded systems that we develop for NASA's Deep Space Network. Generally, the systems developed control receivers, transmitters, exciters, and signal paths through the communication hardware. The most common programming language in our systems is , , 1'1 ./M for Intel 8080, 8086, and 80286 microprocessors; and the systems range in size from 20,000 to 100,000 Non-Commented Lines Of Code (NC] .OC). Approximately 65% of the funding received in our environment is dedicated towards extending the life span of the previously developed systems; of this, 15% is spent on finding and fixing defects, while 8570 is for adding automation features, adding capabilities, and increasing capacity.

Our efforts have been successful in that the life span of our systems arc four to eight years, and increasing. As support for new spacecraft becomes necessary, these older systems arc being used in new ways, thereby increasing the importance of quality, defect free and cost effective enhancements to the. software. Protocols and guidance for locating and rectifying defects in the software sustaining environment were deemed critical especially with the added complications that the people maintaining the systems arc not the people who originally developed them and that there is little or no confidence in the software documentation.

Specifically, wc were looking for ways to identify which modules should be re-engineered, and which modules would need extra development and test time in order to maintain. The problems we face in our environment are quite common in the industry. Software maintenance cost is about two to four times the original development cost, Bochm [3], Glass [13], Fairley [10], and Munson and Khoshgoftoar [9]. Charette [5] emphasizes the fact that 60 to 80 percent of the total software costs arc rc.late.ct to maintenance. This will likely remain so for the indefinite future [7,11, 24].

Figure 1 shows the initial cost breakdown in developing a new project (unfortunately with maintenance costs hidden) and Figure 2 shows the costs of software during its life cycle as discussed by Zelkowitz [34]. Software maintenance is not what people think it is: Software maintenance actually encompasses fixing software errors in addition to software enhancements and adding ncw functions to existing systems, system conversion, training anti supporting users, and improving systems performance [31-33]. Error correction, which is often perceived as the substance of maintenance, is only a small part of the software maintenance effort, Dekleva [8], and Bochm and Papaccio [4]. Table 1 shows the. distribution of the average time spent on various maintenance tasks for four years as reported by Lents and Swanson [19]. Note that functional enhancement constitutes the major portion of the time spent on software maintenance. Charette [5] discusses another reason why the cost of software is so high and cites some statistics as reported by the Comptroller General [6] and shown in Table 2. It is reported that only two percent of the software contracted for could work on delivery, three percent could work after some rework, forty-five percent was delivered but never successfully put to USC, twenty percent was used but either was extensively reworked or abandoned, and thirty percent was paid for but never delivered.

For the project described here.in, we took these steps:

1) determine what the literature suggests;

,2) develop a course of action to be tried on one of our operational systems hopefully as a representative of all the others;

3) perform the steps and analyze the results.

The actions and results of taking each of these steps arc described below.

B. Suggestions from Literature and Course of Action

One of the earlier studies encountered pertaining to our objectives was undertaken by Shen, Yu, Thebaut and Paulsen [27]. This study assessed the potential usefulness of product and process metrics in identifying components of the system which were most likely to contain errors. Their goal was to establish an empirical basis for the usc of objective criteria in developing strategies for the allocation of testing effort in the software maintenance environment. It was found that the number of unique operands, as defined by Halstead [14], was the best predictor of problem reports on modules that were reported after the initial delivery. Additionally, simple mc.tries related to the number of unique operands, such as the cyclomatic complexity (defined by McCabe [20]), also performed well. Shen et. al., concluded that these metrics arc useful in finding error prone modules at an early stage.

Kafura and Reddy [1 7], in 1987, published the results of using software complexity metrics during the software maintenance phase of a system, They related seven separate metrics to the experience of maintenance activities on medium size systems. Two of the results reported were that the overall complexity of a system grows with time and that the individual complexity scores of the software modules agree well with the expert opinions of the programmers. Their conclusion was that metrics could form the control

clement in a formal maintenance method.

Harrison and Cook [15, 16] discusses the decision, frequently encountered by software maintenance personnel, of whether 10 make an isolated change in a module or to totally redesign and rewrite the module anew. They developed an objective decision rule to identify modules which should bc rewritten rather than modified. This decision rule was whether the total change in 1 Halstead Software Science Volume metric exceeded a threshold value. This threshold value seems to bc subjective since it depends upon the risk-taking propensity and experience of the decision maker and must be tuned for a particular environment.

Lennselius, Wohlin and Vrana 18] discuss the possibility of using complexity metrics to identify error-prone modules, and thus maintenance-prone modules. They suggest that a module whose complexity lies at least one standard deviation above the acceptable mean of complexity of the project may bc considered as a maintenance-prone module. The authors however, emphasize that metrics cannot replace the decision-making process of software managers.

Rodriguez and Tsai [23] usc discriminant analysis to develop a methodology for the evaluation of software metrics. The authors suggest that when classifying units of software as either complex or normal, more attention is usually paid to the complex group to either redesign it or test it more thoroughly. '1'heir methodology is based on the assumption of normal distribution and homogeneity of variances of the two groups. The authors considered 13 metrics depicting Halstead's Software Science Metrics, McCabe Complexit y Metrics, and Non-Commented Lines of Code (NCLOC) metrics. They concluded that these metrics arc correlated.

Stalhane [29] discusses how to estimate the number of defects in a software unit from various software metrics, and how to estimate the reliability of the same software. The author also reaches the conclusion that complexity increases as the size of code increases. Stalhane asserts that misunderstanding the specifications will increase with the specification complexity and that complexity may be transferred to the code and thus lead to maintenance-prone complex code and complex modules.

Munson and Khoshgoftoar [21] employ factor analytic techniques to reduce the dimensionality of the complexity problem space to produce a set of reduced metrics. The reduced complexity metrics are subsequently combined into a single relative complexity measure for the purpose of comparing and classifying programs, in particular, the relative complexity metric can be seen to represent the complexity of a particular software module at a particular level of system release.. The authors investigate McCabe Complexity Metrics, Halstead Software Science Metrics and Non-Commented Lines of Code (NCLOC) Metrics. The comparison of complexity is again of a relative and subjective nature.

Binder and Poore [2] investigated the possibility of including the number of comments in the code as a variable in determining the quality of the code. They assert that comments only contribute to quality when they are needed and meaningful. The authors suggest a software quality measure called the "1.11-ratio" defined as the ratio of the number of Operators to the sum of the number of operands and number of comments. The authors agree that their experiments with the "LB-ratio" need additional work and refinement since including the concept of meaningful comments in the formula seems to be problematic and subjective at best.

The suggestions that were deduced from these sources are:

1) An estimate of errors and reliability can be determined from software product metrics [20, 27, 2.9];

2) Software. product metrics could be used to find error prone modules and could form the control element in a formal software maintenance methodology [15-18];

3) The software product metrics that may be considered include all of Halstead's Software Science h4c.tries, McCabe's Complexity Metric [14, 23, 27], and the Non-Comment Source Lines of Code (NCLOC) [21];

4) Factor analysis can be used to identify those software measures that arc highly and significantly related to all other measures, This economy of description will facilitate the analysis of software complexity [21],

5) Comments in the code contribute to the quality of software [2].

We therefore set forth on the following course:

1) Determine the 1 Ialstead Software Science, McCabe Complexity, NCLOC, and LB-Ratio from sequential releases of a representative software system;

2) Perform factor analysis on the metrics from the software modules to determine the unique dimensions represented by the metrics;

3) l'repose a model to calculate a relative metric; and

4) Determine if this metric can identify maintenance prone modules in the software by using the mean plus one standard deviation as the relative metric cutoff value.

11. Method, Analysis and Results

A. Representative System and Metrics Collection

A. 1. Nature of Software

We anal yzed the source program in the Very 1 ong Baseline Interferometry (VI BI) Receiver Controller Software System (VRC), using factor analysis for sixteen software measures. The source program is a real-time embedded system in the receiver-exciter subsystem of NASA's (National Aeronautics and Space Administration) 1 Deep Space Network (lISN). It serves as a communication interface to VLBI subsystems and configures and monitors the status of the Narrow Channel Bandwidth VI BI Receiver Assembly. Three releases of the. system software were analyzed: OP-B (222 modules), OP-C (224 modules), and a draft version of OP-D (235 modules), These were used as a representative maintenance project in this study. The source code for these three releases was originally written in PI ./M, but was late.r converted to C using PLC86 conversion program (from Micro-Processor Services).

A. 2. Software Metrics and Measures.

Software Metrics are quantitative measures of certain characteristics of a development project that can be valuable management and engineering tools. Software metrics can be used to achieve various project-specific results such as: Predicting source-code complexity at the design phase; monitoring and controlling software reliability and functionality, predicting cost and schedule; and identifying high risk modules, in a software project [28].

The sixteen software measures that were used to analyze the VLBI Receiver Controller (VRC) software arc:

1.	n_1	number of unique operators
2.	n_2	number of unique operands
3.	N_1	number of total opc.raters
4.	N_2	number of total operands
5.	N	length (N_1 + N_2)
6.	$\hat{N}$	estimated length = $[n_1(\log_2(n_1))+n_2(\log_2(n_2))]$
7.	v	volume = $N * \log_2(n) = (N_1 + N_2) \log_2(n_1 + n_2)$.
8.	E	effort = $V/[(2/n_1) * (n_2/N_2)]$
9.	VG_1	McCabe Cyclomatic Complexity (number of decisions -t 1)
10.	VG_2	extended complexity (decisions + ANDs 4 ORs -t 1)
11.	1 .(K	lines of code (includes blank and comment lines)
12.	B/C	number of blank lines -1 number of comment lines
13.	<;>	number of executable semi-colons
14.	SP	average maximum lines between variable references
15.	NCI .OC -	Non commented lines of code = 1 .OC - B/C
16.	LB-Ratio -	$[N_1/(N_2+ We)]$

The first eight measures belong to the Halstead software science family of software complexity measures. Halstead [14] uses a series of software science equations to measure the complexity of a program based on the lexical counts of symbols used. Generally, the measurements arc made for each module, and the total measurements of the modules constitute the measurement of the program. Halstead's metrics become available only after the coding is done, and therefore can be of usc only during the testing and maintenance phases. Although Halstead's metrics are useful in determining the complexity of programs, their weaknesses arc that they do not measure control flow complexity, and have

little predictive value.

Measures number nine and ten, i.e. $VG1$ and $VG2$ belong to McCabe and were adapted from the mathematical concepts of graph theory. McCabe cyclomatic complexity metric $VG1$ is a measure of the maximum number of linearly independent circuits in a program control graph. The primary purpose of this metric is to identify software modules that will be difficult to test or maintain as explained by McCabe [20]. The value of McCabe metric is available only after the detailed design is done. Although McCabe metric is very useful at measuring control flow complexity, its weakness is that it is not sensitive to program size; for example, if programs of different size are composed exclusively of sequential statements, then they may have the same cyclomatic number.

Measures number eleven to fifteen deal with the size of the program or number of lines. Although many researchers do not find this measure as appealing, Boehm [3] points out that no other metric has a clear advantage over NCLOC as a metric. It is easy to measure, conceptually familiar to software developers, and it is used in most productivity databases and cost estimation models.

Measure number sixteen, the "LB-Ratio"; is defined by Binder and Poore [2] as the ratio of the number of operators to the sum of the number of operands and number of comments. It appears to capture the idea of distinguishing between meaningful comments in the code and just comments in general. The weakness of this metric is its reliance on defining the number of meaningful comments which seems to be more subjective than quantitative.

B. Analysis of' Data, Models, and Validation

The sixteen software measures of the three releases of the (VRC) code; (OP-B, OP-C and draft OP-D) were analyzed using Factor Analysis, Correlation, Analysis of Variance and Regression Analysis. Table 3 shows the number of modules and the mean value per module for each of the sixteen measures. Tables 4-6 show the correlation matrix of the sixteen measures for the three releases. The data show a high degree of correlation. Except for the measure "1 B-Ratio", the remaining fifteen measures are highl y correlated. It can be seen that the Halstead volume metric (V), McCabe Cyclometric Complexity metric (VG 1) and NCLOC metric are highly and significantly correlated while the I B-Ratio metric is not. '] 'hese results agree with what other researchers have found, Ramamurthy and Melton [22], Gill and Kemerer [12], Samadzadeh and Nandakumar [25], Basili and Hutchins [1], Evangelist [9] and Kafura and Reddy [17].

The factor analysis matrix is shown in I'able 7. All measures except the I B-Ratio are. loaded on factor 1, and thus there is no cross-loading, This is a desired result, since cross-loading on many factors makes the interpretation of the result ambiguous. The Analysis of Variance of the three sets of rc.leases did not show any significant difference at the level of significance of 0.05. This means that on the average the values of say, the McCabe Cyclomatic Complexity Metric (VG1) of the three releases arc not significantly different at Alpha of 5%. The same is also true for the other fifteen measure.s.

Regression Analysis had been used to develop models of relationships of the most interrelated measures. These arc: The Halstead Volume Metric (V), the McCabe Cyclomatic Metric (VG 1), and the Non Commented Lines of Code (NCLOC) metric, as discussed next.

B.1. Factor Analysis Discussion

Three releases of software were analyzed by factor analysis to show the existence of meaningful relationships among known software complexity measures. The analysis shows the number of factors where software complexity measures tend to load high or low, and also the percentage of the. variability explained by each factor. This research also shows the matrix of correlation summarizing the relationships among the sixteen software complexity measures for each release.

Factor analysis of the three releases of software had shown that the first fifteen measures of complexity are closely related to some measure of similarity and are in consequence all interrelated. However, the sixteenth complexity measure (1.13-Ratio) does not seem to be typical of the other fifteen measures, and thus it is unlike the rest of the data set. The three releases show two factors that concisely state the pattern of relationships within the sixteen measures. however, measures one to fifteen load most strongly on the first factor with explained variability of 90% to91 %, while the second factor displays less interesting patterns with loading of 9% to 10%. Factor analysis had also shown that three complexity measures: the McCabe Cyclomatic Complexity Metric (VG1), the Halstead Volume Metric (V), and (NCLOC) are highly and strongly related. Therefore, in order to achieve an economy of description, these three measures are considered to give a strong similarity and representation of all the fifteen measure.s.

The correlation matrix for each release of the software also shows that the first fifteen complexity measures are related, while the LB-Ratio measure is not related or interrelated to any of the other fifteen measures.

Analysis of variance does not show any significant difference between the three

releases at the level of significance of 5%. this means that as the software evolves through its releases, the interrelationships between the complexity measures seem to be preserved. However we should note that without normalization to size, adding on to a program will make a more complex program. This seems to agree with what other researchers have found as discussed by Valett and McGarry [30], Harrison and Cook [15] and Schneidewind [26].

Since factor analysis techniques showed that the first fifteen software measures are closely related to some measure of similarity, and since three of these measures: McCabe Cyclomatic Complexity Metric (VG 1), Halstead Volume Metric (V), and NCLOC metric are highly and significantly related, they are considered to give a strong similarity and representation of all fifteen measures. This economy of description made it appealing to develop a set of parsimonious models for software complexity measurements using data from the three software releases. The five composite models together with their coefficient of determination (R^2) are as follows:

1. $\langle VG1 \rangle = 1.48 + 0.005(V)$, $R^2=96\%$
2. $\langle VG1 \rangle = 0.510 + 0.136(NCLOC)$, $R^2=96\%$
3. $\langle VG1 \rangle = 0.786 + 0.001\ 3(V) + 0.0976\ (NCLOC)$, $R^2=96\%$
4. $\langle v \rangle = -206 + 29.5(NCLOC)$, $R^2=99\%$
5. $\langle v \rangle = -210 + 8.7(VG1) - t\ 28.3(NCLOC)$, $R^2=99\%$

Statistical analysis, model back testing, and model testing with independent segments of software are used for validation of the composite models and ascertaining their degree of accuracy. The developed models had shown a high degree of accuracy in predicting software complexity and thus they can serve as baseline for other software projects in identifying software modules with high complexity (maintenance prone) so that actions can be taken before their release to users.

B.2. Back Testing of Models

The five composite complexity models shown above were checked with actual data from the three releases, OP-B, OP-C and OP-D. Table 8 and Figure 3 show the actual average values of the dependent variables (VG1) and values predicted by the first three models. 'l'able 9 and Figure 4 show the actual average values of (V) and values predicted by models 4 and 5. It can be seen that the difference in predicting (VG 1) by the first three composite models ranges from 3.2% to 10.6% below actual average value of (VG1) as calculated by McCabe Cyclomatic Complexity metric. Also, the difference in predicting (V) by models four and five ranges from 1.2% to 1.3% above act ual average value of (V) as calculated by Halstead's Volume. metric.

B. 3. Testing The Five Composite Models by External Check

The five composite complexity models Were tested against fOUr independent segments of software with characteristics as shown in '1'able 10. A sample calculation of actual average values of (VG 1) ant] values predicted by Model1 for the four segments of software is shown in 'I'able 11. The summary of the actual grand average values of (VG 1) and (V) and their values as predicted by Models 1 ,2,3 and Models 4 and S respectively for the four segments of software is shown in Table 12, and 13 and Figures 5 and 6. It can be seen that the difference in predicting (VG 1) by the first three composite models ranges from 17.3% below to 0.7% above actual average value of (VG 1). Also, the difference in predicting (V) by models four and five is 9,7% above actual avc.rage value of (V) for the four segments of software.

C. Parsimonious Model and Representative System

Since the five complexity models developed in this study show direct relationships between (VG1) and (V) and also (NCLOC); wc had chosen the third model

$$<VG1> = 0.786 + 0.0013(V) + 0.0976(NCLOC)$$

as a representative model for estimating the value of (VG1) given the measured values of (V) and (NCLOC).

C. 1. Development of the Relative Complexity Metric

We propose to capture the total complexity of a program based on its control flow complexity, the lexical counts of symbols used, and the program sire. In essence, a complexity metric that accounts for a program total complexity due to volume and control flow and normalized by the number of lines of code would present a relative complexity metric that is more useful to consider for detecting maintenance-prone programs. The relative complexity metric (RCM) will be derived for each module from the measured value of (V), the estimated value of (VG1) from model 3, and normalized by the module lines of code. The RCM for module is:

$$(RCM)_i = \left(\frac{VG_1 + V}{NCLOC}\right)_i$$

C. 2. Analysis of The Three Releases Using The Relative Complexity Metric (RCM)

The Relative Complexity Metric (RCM) was used to analyze the modules of the three releases as shown below.

Release	Total # Of Modules	Relative Complexity					
		Total	Max	Min	Median	Mean	std. Dev.
OP-B	222.	2799	45	0.4	10.9	12.6	10.0
OP-C	224	2837	45	0.4	10.9	12.7	9.6
01'-1)	235	3470	49	- 0.4	12.2	14.8	11.3

Note that, as reported by Kafura and Reddy [17], the Relative Complexity Metric (RCM) has grown with each release from a 2799 total in 0P-B to a 3470 total in the draft of OP-D.

Using the criterion of The mean relative. complexity value plus onc standard deviation as a cut-off value for acceptable modules, wc can identify those modules that can bc considered as outliers, or mai ntenance-prone modules. Wc obt ain the following for the three releases:

Release	Total #of Modules	(RCM) Cutoff Value	#of Modules Exceeding (RCM) Cutoff Value	% Modules over (RCM) Cutoff Value
OP-B	222	22..6	33.0	15.0
OP-C	224	22.3	36.0	16.0
OP-D	235	26.1	35.0	15.0

in order to determine whether the modules above the cutoff value were more at risk (o be modified for enhancement or fixes than modules below the cutoff value, the transitions between the re.leases were examined. The results appear in the table below. Of the 33 modules over the cutoff value of RCM in 01'-11, 40% were actually modified in order to implement OP-C. Of the 36 modules in OP-C over OP-C's RCM cutoff value, SO% were actual] y modified to implement the draft version of OP-D.

Transit ion	# of Modules Modified	(RCM) Cutoff Value	% of Modified Modules Over Cutoff Value	% of all Modules Over Cutoff Value that were Actually Modified
1 'rom OP-B to OP-C	13	22.6	46	40
1 'rom OP-C to OP-D	38	22.3	47	50

Although the cutoff value seems to evenly divide the modules that were actually modified, the modules over the cutoff value for each release were more likel y (o bc changed than the modules below the cutoff value. The relative complexity metric (RCM) was therefore able to identify maintenance prone modules.

111. Discussion and Conclusion

Given that a metric which measures software complexity s}1ouIc1 prove to be a useful predictor of software maintenance costs, it is recommended that modules that show a high order of

complexity within a release be looked upon as modules with propensity to become maintenance prone after release and delivery to users. It is imperative that a maintenance prone module be improved, enhanced, or simplified into two or more modules before final delivery. The composite complexity models and the relative complexity metric developed in this study can be considered as a baseline for comparison with other projects and may serve as a set point for simplifying and reducing complexity of developed software.

Acknowledgment

This research was carried out by the Jet Propulsion Laboratory, California Institute of Technology under contract with the National Aeronautics and Space Administration. The authors would like to express their sincere thanks to Dr. William J. Hurd, Deputy Manager, and Paul A. Willis; Supervisor, Radio Frequency and Microwave Subsystems Section, Dr. Robert C. Tausworthe; Chief Technologist, Information Systems Division, and Dr. Donald S. Remer, Telecommunications and Data Acquisition Planning for comments and suggestions that greatly improved this report.

REFERENCES

[1] V. R. Basili and D. H. Hutchins, "An Empirical Study of a Synthetic Complexity Family," IEEE Trans. Software Engineering, 9, pp. 664-672, 1983

[2] L. H. Binder and J. M. Poore, "Field Experiments with Local Software Quality Match-its," Software Practice and Experience, 20, pp. 631-647, 1990

[3] B. Boehm, Software Engineering Economics, Prentice Hall, Englewood Cliffs, N.J., 1981

[4] B. Boehm and P. Papaccio, "Understanding and controlling Software Costs," IEEE Trans. Software Engineering, 14, pp. 1462-1477, 1988

[5] R. N. Charette, Software Engineering Environment, McGraw Hill, Inc., New York, N. Y., 1986

[6] Comptroller General, Contracting For Computer Software Development, General Accounting Office Report, GAO, FGMSD-80-4, 1979

[7] B. Curtis, S. Sheppard, P. Milliman, M. Borst and T. Love, "Measuring the Psychological Complexity of Software Maintenance Tasks With The Halstead and McCabe Metrics," IEEE Trans. Software Engineering, 5, pp. 96-104, 1979

[8] S. Dekleva, "Software Maintenance: Any News Besides The Name," The Software Practitioner, 3, pp. 5-8, 1993

[9] W. M. Evangelist, "Software Complexity Metric Sensitivity to Program Structure Rules," J. of Systems and Software, 3, pp. 231-243, 1983

[10] R. E. Fairley, Software Engineering Concepts, McGraw Hill, New York, N. Y., 1985

[11] V. R. Gibson and J. A. Senn, "System Structure and Software Maintenance Performance," Communications ACM, 32, pp. 347-358, 1989

[12] G. K. Gill and C. F. Kemerer, "Cyclomatic Complexity Density and Software Maintenance Productivity," IEEE Trans. Software Engineering, 17, pp. 1284-1288, 1991

[13] R. 1.. Glass, Software Maintenance Handbook, Englewood Cliffs, N, J., Prentice I la]], 1981

[14] M. Halstead, Elements of Software Science, New York, N.Y. Elsevier North Holland, Inc., 1977.

[15] W, Harrison and C. Cook, "A Micro/Macro Measure of Software Complexity," The Journal of Systems and Software, 7, pp. 213--219, 1987

[16] W. Harri son and C. Cook, Insights on Improving The Maintenance Process Through Software Measurements. Report Naval Ocean Systems Center, TR 90-4, N66001-87-D-0136, 1990.

[17] D. Kafura and G. R. Reddy, "The Use of Software Complexity Metrics in Software Maintenance," IEEE Trans. Software Engineering, 13, pp. 335-343, 1987

[18] B. Lennselius C. Wohlin and C. Vrana, "Software Metrics: Fault Content Estimation and Software Process Control," Microprocessors and Microsystems, 11, pp. 365-375, 1987.

[19] B. P. Lientz and E. B. Swanson, Software Maintenance Management, Reading, MA. "Addison-Wesley, 1980.

[20] 'J'. J. McCabe, "A complexity Measure," IEEE Trans. Software Engineering, 2, pp. 308-32.0, 1976.

[21] C. Munson, and T.M. Khoshgoftaar, "Application of a Relative Complexity Metric For Software Project Management," Journal of Systems and Software, 12, pp. 283-291, 1990

[2.2] B. Ramamurthy and A, Melton, "A Synthesis of Software Sciences Measures and the Cyclomatic Number," IEEE 'J'rans. .S~fl.ware Engineering, 14, pp. 1116-1121, 1988

[23] V. Rodriguez, and W. T. Tsai, "Evaluation of Software Metrics Using Discriminant Analysis," The Eleventh Ann. Int. Computer Software and Applications Conf., pp. 245-251, 1987.

[24] H. D. Rombach, "A Controlled Experiment On The impact of Software Structure on Maintainability," IEEE Trans. Software Engineering, 13, pp. 344-354, 1987

[25] M. H. Samadzadeh and K. Nandakumar, "A Study of Software Metrics," J. Systems Software, 16, pp. 229-234, 1991

[26] N.F. Schneidewind, "Methodology For Validating Software Metrics," IEEE Trans. Software Eng. 18, pp. 410-422, 1992.

[27] V. Y. Shen, T. Yu, Sm. M. Thebaut and L. R. Paulsen, "Identifying Error-Prone Software-An Empirical Study," IEEE Trans. S/W Eng., 11, pp. 317-323, 1985

[28] Y. S. Sherif, E. Ng and J. Steinbacher," Computer Software Development: Quality Attributes, Measurements and Metrics," Naval Research Logistics, 35, pp. 425-436, 1988

[29] T. Stalhane, A Discussion of Software Metrics As A Mean For Software Reliability Evaluation. Report # PB89-210322, U.S. Dept. of Commerce, National Technical information Service, 1988.

[30] J.D. Valett and F.E. McGarry, "A Summary of Software Measurement Experiences in the Software Engineering Laboratory," The Journal of System and Software, 9, pp. 137-148, 1989

[31] 1. Vessey and R. Weber, "Some Factors Affecting Program Maintenance: An Empirical Study," Communications ACM, 26, 2, pp. 128-134, 1983

[32] S. Wake, and S. Henry, "A Model Based on Software Quality Factors Which Predicts Maintainability," Proceedings, Conference on Software Maintenance, Phoenix, Arizona, Ott. 24, 1988, pp. 382-387, 1988.

[33] S. S. Yau and J. S. Collofello, "Sonic Stability Measures For Software Maintenance," IEEE Trans. Software Engineering, 6, pp. S4S-552., 1980

[34] M. V. Zelkowitz, A. C. Shaw and J. D. Grannon, Principles of Software Engineering and Design, Prentice Hall, Inc., Englewood Cliffs, N. J, 1979

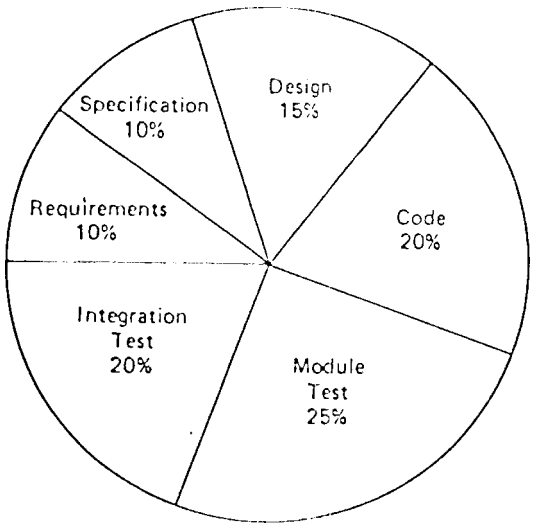

Figure 1. The initial Cost Breakdown in Developing a New Project

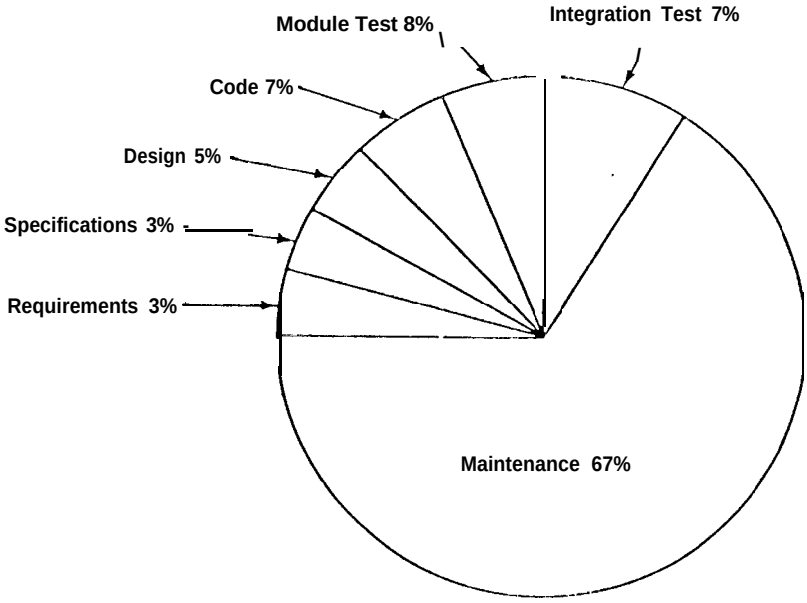

Figure 2. The Cost of' Software During its Life Cycle

Table 1. Percentage of Time Spent On Various Maintenance Tasks

	% Time Spent			
Maintenance Tasks	1977	1985	1987	1990
Enhancements	59	44	41	43
Corrections	22	15	18	16
Supporting Users	NA	21	12	12
Reengineering	NA	NA	10	9
Adaptations	6	8	9	8
Documentation	6	NA	5	6
Tuning	4	NA	3	5
Evaluating Requests	NA	8	NA	NA
Other	3	4	2	1

‘1’able 2. Comptroller **General** Statistics on **Delivered** Software

	Quality of Software **Delivered**	Percentage (%)) of Software **Delivered**
1.	Could work 011 delivery	2
2.	Could work after some rework	3
3.	Never successfully put to usc	45
4.	Extensivel y reworked	20
5.	Useless	30
	Total	100

Table 3. 01'-11, **OP-C and OP-D** Modules **and** Mean **Value** of the Sixteen **Measures**

Measure	OP-B (222 Modules) Mean	OP-C (224 Modules) Mean	OP-D (23s Modules) Mean
1. n1	12	12	13
2. 112	12	12	15
3 N]	70	75	87
4. N2	42	44	52
5. N	113	119	140
6. $\hat{N}$	103	110	126
7. V	704	721	844
8. E	53781	58198	61715
9. VG1	4	4	5
10, VG2	5	4	5
11. LOC	73	78	83
12. B/C	43	46	49
13. <;>	12	13	15
14. SP	5	5	6
15. NCLOC	30	31	34
16. LB-Ratio	1	1	1

Table 4. Correlation Matrix of Sixteen Measures For OP-B

Pearson Correlation Coefficients / Prob > \|R\| under Ho: Rho=0 / N = 222

	B1	B2	N1	N2	N	h'V	v	E	VG1	VG2	LOC	BC	CR	SP	NLOC	LBR
B1	1.00000 0.0	077205 0.0001	0.70872 0.0001	069055 0.0001	0.70054 0.0001	o 72123 o 0001	0.65058 0.0001	0.54830 0.0001	0.74993 0.0001	o 56794 0.0001	0.7172? 0.0001	0.59043 0.0001	0.76131 0.0001	0.71629 0.0001	0.73042 0.0001	-0.03466 0.6075
B2	o 77205 0.0001	1.00000 0 0	0.91155 0.0001	o 93063 0.0001	0.9223? 0.0001	0 96182 0.0001	O 91273 0.0001	0.78438 0.0001	0.93249 0.0001	0.73144 0.0001	0.87705 0.0001	0.58032 0.0001	0936$4 0.0001	0.62632 0 0001	o 91476 o 0001	-0.03778 0.5755
N1	O 70872 0.0001	0 91155 0.0001	1.00000 0.0	0 98580 0.0001	0.99801 0. 001	0 99913 0.0001	0.98176 0.0001	0.95128 0.0001	0.93309 0.(XXJI	O 71636 0 0001	0.89288 0.0001	0.51917 0.(XKI!	0 95732 0 000 1	0.62622 o 0001	0 97883 0 0001	0.01953 0.7723
N2	0 69055 o 0001	0.93063 o 0001	0.98580 0.0001	I 00000 0.0	0.99437 0.0001	0 99243 0.0001	0.97363 0 0001	0.99750 0.0001	0.89624 0.0001	0.71224 0.0001	0.91111 0 0001	0.56573 0.0001	095695 o 0001	0.6! 197 0.0001	0.97494 0.0001	-0.00079 0.9906
N	0.70054 0.0001	O 92233 0.0001	0.99801 0.0001	099437 0.0001	1.00000 0.0	0.92114 0.0001	o 92277 o 0001	0.92915 0.'cCCI	0.92274 0.0001	0.72116 0.0001	0.902s0 0.0001	o 53733 0.0001	096022 0.0001	0.62291 0.0001	0.98104 0.0001	0.01181 o 8611
NV	0.72122 0.0001	0.96182 0.0001	0.99913 0.0001	0 93243 0.0001	0.92114 0.0001	1 00000 0.0	0.90741 0.0001	0.79853 0.0001	0.76636 0.0001	0.64927 0.0001	0.85885 0.0001	0.57506 0.0001	0.88360 0.0001	0.47706 0.0001	0.89127 0.0001	-0.02969 0.6600
v	0.65058 0.0001	0 91273 0 0001	0.98176 0 0001	0.97363 0.0001	0.98277 0 0001	0 90741 0.0001	1 00000 0 0	0.95253 0.0001	0.91233 0.0001	0 79787 0.0001	0.86609 0.0001	0.47319 0.0001	0 93203 0 0001	0.58001 0.0001	0.96965 0.0001	0.01466 0.8281
E	0.54830 0.0001	0.78438 0.0001	0.95128 0.0001	0.99250 0.0001	0.93915 0.0001	0.79853 0.0001	0.95253 0.0001	1.00000 0 0	0.89454 0.0001	0.69025 0.0001	0.78596 0.0001	0.37656 0.0001	0.85869 0.0001	0.49117 0.0001	0.91505 0.0001	0.04023 0.5510
VG1	0.74993 0.0001	0.93249 0.0001	0.93309 0.0001	0.89624 0.0001	0.92274 0.0001	0.76636 0.0001	0.91233 0.0001	0.89454 0.0001	1.00000 0 0	0.78359 0.0001	0.86099 0.0001	0.48352 0.0001	0.92553 0.0001	0.76376 0.0001	0.95509 0.0001	0.01807 0.7959
VG2	0.56794 0.0001	0.73144 0.0001	0.71636 0.0001	0.71234 0.0001	0.72116 0.0001	0.64927 0.0001	0.79787 0.0001	0.69025 0.0001	0.78359 0.0001	1.00000 0 0	0.67531 0.0001	0.32942 0.0001	0.70748 0.0001	0.58539 0.0001	0.78232 0.0001	-0.02037 0.7628
LOC	0.75723 0.0001	0.87705 0.0001	0.89288 0.0001	0.91111 0.0001	0.90250 0.0001	0.85885 0.0001	0.86609 0.0001	0.78596 0.0001	0.86099 0.0001	0.67531 0.0001	1.00000 0 0	0.83221 0.0001	0.90148 0.0001	0.65696 0.0001	0.92972 0.0001	-0.15599 0.0201
BC	0.59043 0.0001	0.58032 0.0001	0.51917 0.0001	0.56573 0.0001	0.53733 0.0001	0.57506 0.0001	0.47319 0.0001	0.37656 0.0001	0.48352 0.0001	0.32942 0.0001	0.83221 0.0001	1.00000 0.0	0.56689 0.0001	0.43891 0.0001	0.56954 0.0001	-0.33967 0.0001
CR	0.76131 0.0001	0.93684 0.0001	0.95732 0.0001	0.95695 0.0001	0.96022 0.0001	0.88360 0.0001	0.93203 0.0001	0.85869 0.0001	0.92553 0.0001	0.70748 0.0001	0.90148 0.0001	0.56689 0.0001	1.00000 0 0	0.75125 0.0001	0.95988 0.0001	0.01323 0.8445
SP	0.71629 0.0001	0.63632 0.0001	0.62622 0.0001	0.61197 0.0001	0.62291 0.0001	0.47706 0.0001	0.58001 0.0001	0.49117 0.0001	0.76376 0.0001	0.58539 0.0001	0.65696 0.0001	0.43891 0.0001	0.75125 0.0001	1.00000 0.0	0.68239 0.0001	0.01060 0.8752
NLOC	0.73042 0.0001	0.91476 0.0001	0.97883 0.0001	0.97494 0.0001	0.98104 0.0001	0.89127 0.0001	0.96965 0.0001	0.91505 0.0001	0.95509 0.0001	0.78232 0.0001	0.92972 0.0001	0.56954 0.0001	0.95988 0.0001	0.68239 0.0001	1.00000 0.0	-0.00566 0.9332
LBR	-0.01466 0.6075	-0.03778 0.5755	0.01953 0.7723	-0.00079 0.9906	0.01181 0.8611	-0.02969 0.6600	0.01466 0.8281	0.04023 0.5510	0.01807 0.7889	-0.02037 0.7628	-0.15599 0.0201	-0.33967 0.0001	0.01323 0.8445	0.01060 0.8752	-0.00566 0.9332	1.00000 0.0

Table 5. Correlation Matrix of Sixteen Measures For OP-C

Pearson Correlation Coefficients / Prob > |R| under Ho: Rho=0 / N = 224

	B1	B2	N1	N2	N	M	V	E	VG1	VG2	LOC	BC	CT	SP	NCLOC	LDR
B1	1.00000 0.0	0.79503 0.0001	0.70854 0.0001	0.68603 0.0001	0.70381 0.0001	0.85592 0.0001	0.66948 0.0001	0.54823 0.0001	0.75377 0.0001	0.75702 0.0001	0.74387 0.0001	0.62827 0.0001	0.77626 0.0001	0.68827 0.0001	0.72105 0.0001	-0.04119 0.5397
B2	0.79503 0.0001	1.00000 0.0	0.91747 0.0001	0.92764 0.0001	0.92585 0.0001	0.98457 0.0001	0.90915 0.0001	0.78541 0.0001	0.83939 0.0001	0.83961 0.0001	0.89238 0.0001	0.68225 0.0001	0.95180 0.0001	0.56033 0.0001	0.90922 0.0001	-0.01885 0.7790
N1	0.70854 0.0001	0.91747 0.0001	1.00000 0.0	0.98567 0.0001	0.99822 0.0001	0.92925 0.0001	0.99735 0.0001	0.95202 0.0001	0.93489 0.0001	0.93273 0.0001	0.93188 0.0001	0.64823 0.0001	0.95255 0.0001	0.49795 0.0001	0.98920 0.0001	0.02460 0.7143
N2	0.68603 0.0001	0.92764 0.0001	0.98567 0.0001	1.00000 0.0	0.99326 0.0001	0.91839 0.0001	0.98180 0.0001	0.90939 0.0001	0.89666 0.0001	0.89658 0.0001	0.93627 0.0001	0.67809 0.0001	0.95307 0.0001	0.49197 0.0001	0.97728 0.0001	0.00933 0.8896
N	0.70381 0.0001	0.92585 0.0001	0.99822 0.0001	0.99326 0.0001	1.00000 0.0	0.92967 0.0001	0.99510 0.0001	0.93890 0.0001	0.92500 0.0001	0.92352 0.0001	0.93758 0.0001	0.66283 0.0001	0.95701 0.0001	0.50224 0.0001	0.98868 0.0001	0.01943 0.772
M	0.85592 0.0001	0.98457 0.0001	0.92925 0.0001	0.91839 0.0001	0.92967 0.0001	1.00000 0.0	0.91803 0.0001	0.81228 0.0001	0.88911 0.0001	0.88833 0.0001	0.89454 0.0001	0.66532 0.0001	0.96145 0.0001	0.61427 0.0001	0.92292 0.0001	-0.00182 0.9783
V	0.66948 0.0001	0.90915 0.0001	0.99735 0.0001	0.98180 0.0001	0.99510 0.0001	0.91803 0.0001	1.00000 0.0	0.96468 0.0001	0.92532 0.0001	0.92202 0.0001	0.92287 0.0001	0.63301 0.0001	0.94184 0.0001	0.46109 0.0001	0.98518 0.0001	0.02808 0.6760
E	0.54823 0.0001	0.78541 0.0001	0.95202 0.0001	0.90939 0.0001	0.93890 0.0001	0.81228 0.0001	0.96468 0.0001	1.00000 0.0	0.89747 0.0001	0.89059 0.0001	0.85331 0.0001	0.54267 0.0001	0.84510 0.0001	0.33267 0.0001	0.93731 0.0001	0.04221 0.5287
VG1	0.75377 0.0001	0.83939 0.0001	0.93489 0.0001	0.89666 0.0001	0.92500 0.0001	0.88911 0.0001	0.92532 0.0001	0.89747 0.0001	1.00000 0.0	0.99669 0.0001	0.89309 0.0001	0.60974 0.0001	0.92239 0.0001	0.65413 0.0001	0.95515 0.0001	0.02546 0.7047
VG2	0.75702 0.0001	0.83961 0.0001	0.93273 0.0001	0.89658 0.0001	0.92352 0.0001	0.88833 0.0001	0.92202 0.0001	0.89059 0.0001	0.99669 0.0001	1.00000 0.0	0.89196 0.0001	0.61315 0.0001	0.91863 0.0001	0.66041 0.0001	0.95136 0.0001	0.02426 0.7180
LOC	0.74387 0.0001	0.89238 0.0001	0.93188 0.0001	0.93627 0.0001	0.93758 0.0001	0.89454 0.0001	0.92287 0.0001	0.85331 0.0001	0.89309 0.0001	0.89196 0.0001	1.00000 0.0	0.87132 0.0001	0.91200 0.0001	0.50984 0.0001	0.95278 0.0001	-0.12536 0.0611
BC	0.62827 0.0001	0.68225 0.0001	0.64823 0.0001	0.67809 0.0001	0.66283 0.0001	0.66532 0.0001	0.63301 0.0001	0.54267 0.0001	0.60974 0.0001	0.61315 0.0001	0.87132 0.0001	1.00000 0.0	0.65911 0.0001	0.37603 0.0001	0.68116 0.0001	-0.31940 0.0001
CT	0.77626 0.0001	0.95180 0.0001	0.95255 0.0001	0.95307 0.0001	0.95701 0.0001	0.96145 0.0001	0.94184 0.0001	0.84510 0.0001	0.92230 0.0001	0.91863 0.0001	0.91200 0.0001	0.65911 0.0001	1.00000 0.0	0.64830 0.0001	0.95282 0.0001	0.01914 0.7757
SP	0.68827 0.0001	0.56033 0.0001	0.49795 0.0001	0.49197 0.0001	0.50224 0.0001	0.61427 0.0001	0.46109 0.0001	0.33267 0.0001	0.65413 0.0001	0.66041 0.0001	0.50984 0.0001	0.37603 0.0001	0.64830 0.0001	1.00000 0.0	0.52798 0.0001	0.02198 0.7435
NCLOC	0.72105 0.0001	0.90922 0.0001	0.98920 0.0001	0.97728 0.0001	0.98868 0.0001	0.92292 0.0001	0.98518 0.0001	0.93731 0.0001	0.95515 0.0001	0.95136 0.0001	0.95278 0.0001	0.68116 0.0001	0.95282 0.0001	0.52798 0.0001	1.00000 0.0	0.01062 0.8744
LDR	-0.04119 0.5397	-0.01885 0.7790	0.02460 0.7143	0.00933 0.8896	0.01943 0.772	-0.00182 0.9783	0.02808 0.6760	0.04221 0.5287	0.02546 0.7047	0.02426 0.7180	-0.12536 0.0611	-0.31940 0.0001	0.01914 0.7757	0.02198 0.7435	0.01062 0.8744	1.00000 0.0

Table 6. Correlation Matrix of Sixteen Measures For OP-D

Pearson Correlation Coefficients / Prob > |R| under Ho: Rho=0 / N = 235

	B1	B2	N1	N2	N	M	V	E	VG1	VG2	LOC	BC	CT	SP	NCLOC	LBR
B1	1.00000 0.0	0.76798 0.0001	0.69921 0.0001	0.67628 0.0001	0.69470 0.0001	0.83404 0.0001	0.65911 0.0001	0.54051 0.0001	0.70251 0.0001	0.70917 0.0001	0.75883 0.0001	0.65341 0.0001	0.73748 0.0001	0.70268 0.0001	0.72216 0.0001	-0.06819 0.2979
B2	0.76798 0.0001	1.00000 0.0	0.91318 0.0001	0.92727 0.0001	0.92209 0.0001	0.98371 0.0001	0.90476 0.0001	0.76771 0.0001	0.81731 0.0001	0.82739 0.0001	0.88841 0.0001	0.67125 0.0001	0.93009 0.0001	0.62854 0.0001	0.90618 0.0001	-0.02990 0.6484
N1	0.69921 0.0001	0.91318 0.0001	1.00000 0.0	0.98554 0.0001	0.99823 0.0001	0.93052 0.0001	0.99630 0.0001	0.94063 0.0001	0.89809 0.0001	0.89640 0.0001	0.91856 0.0001	0.62890 0.0001	0.94779 0.0001	0.62333 0.0001	0.97910 0.0001	0.02353 0.7197
N2	0.67628 0.0001	0.92727 0.0001	0.98554 0.0001	1.00000 0.0	0.99310 0.0001	0.92374 0.0001	0.98136 0.0001	0.89946 0.0001	0.86324 0.0001	0.86590 0.0001	0.92109 0.0001	0.65207 0.0001	0.94822 0.0001	0.61295 0.0001	0.96792 0.0001	0.00753 0.9086
N	0.69470 0.0001	0.92209 0.0001	0.99823 0.0001	0.99310 0.0001	1.00000 0.0	0.93185 0.0001	0.99426 0.0001	0.92832 0.0001	0.88867 0.0001	0.88864 0.0001	0.92391 0.0001	0.64142 0.0001	0.95138 0.0001	0.62209 0.0001	0.97908 0.0001	0.01824 0.7809
M	0.83404 0.0001	0.98371 0.0001	0.93052 0.0001	0.92374 0.0001	0.93185 0.0001	1.00000 0.0	0.92089 0.0001	0.80819 0.0001	0.85805 0.0001	0.86369 0.0001	0.90027 0.0001	0.66906 0.0001	0.93851 0.0001	0.67833 0.0001	0.92550 0.0001	-0.01502 0.8189
V	0.65911 0.0001	0.90476 0.0001	0.99630 0.0001	0.98136 0.0001	0.99426 0.0001	0.92089 0.0001	1.00000 0.0	0.95835 0.0001	0.88574 0.0001	0.88249 0.0001	0.90801 0.0001	0.61560 0.0001	0.93267 0.0001	0.58713 0.0001	0.97180 0.0001	0.02684 0.6823
E	0.54051 0.0001	0.76771 0.0001	0.94063 0.0001	0.89946 0.0001	0.92832 0.0001	0.80819 0.0001	0.95835 0.0001	1.00000 0.0	0.83132 0.0001	0.81862 0.0001	0.82410 0.0001	0.52488 0.0001	0.82351 0.0001	0.48411 0.0001	0.90390 0.0001	0.04260 0.5158
VG1	0.70251 0.0001	0.81731 0.0001	0.89809 0.0001	0.86324 0.0001	0.88867 0.0001	0.85805 0.0001	0.88574 0.0001	0.83132 0.0001	1.00000 0.0	0.99307 0.0001	0.86436 0.0001	0.55703 0.0001	0.92329 0.0001	0.78499 0.0001	0.94385 0.0001	0.02552 0.6971
VG2	0.70917 0.0001	0.82739 0.0001	0.89640 0.0001	0.86590 0.0001	0.88864 0.0001	0.86369 0.0001	0.88249 0.0001	0.81862 0.0001	0.99307 0.0001	1.00000 0.0	0.86403 0.0001	0.56309 0.0001	0.92001 0.0001	0.79418 0.0001	0.93942 0.0001	0.02379 0.7168
LOC	0.75883 0.0001	0.88841 0.0001	0.91856 0.0001	0.92109 0.0001	0.92391 0.0001	0.90027 0.0001	0.90801 0.0001	0.82410 0.0001	0.86436 0.0001	0.86403 0.0001	1.00000 0.0	0.86829 0.0001	0.90087 0.0001	0.64902 0.0001	0.94701 0.0001	-0.13903 0.0332
BC	0.65341 0.0001	0.67125 0.0001	0.62890 0.0001	0.65207 0.0001	0.64142 0.0001	0.66906 0.0001	0.61560 0.0001	0.52488 0.0001	0.55703 0.0001	0.56309 0.0001	0.86829 0.0001	1.00000 0.0	0.61517 0.0001	0.43323 0.0001	0.66295 0.0001	-0.33398 0.0001
CT	0.73748 0.0001	0.93009 0.0001	0.94779 0.0001	0.94822 0.0001	0.95138 0.0001	0.93851 0.0001	0.93267 0.0001	0.82351 0.0001	0.92329 0.0001	0.92001 0.0001	0.90087 0.0001	0.61517 0.0001	1.00000 0.0	0.75191 0.0001	0.96130 0.0001	0.01875 0.7749
SP	0.70268 0.0001	0.62854 0.0001	0.62333 0.0001	0.61295 0.0001	0.62209 0.0001	0.67833 0.0001	0.58713 0.0001	0.48411 0.0001	0.78499 0.0001	0.79418 0.0001	0.64902 0.0001	0.43323 0.0001	0.75191 0.0001	1.00000 0.0	0.69901 0.0001	0.02089 0.7500
NCLOC	0.72216 0.0001	0.90618 0.0001	0.97910 0.0001	0.96792 0.0001	0.97908 0.0001	0.92550 0.0001	0.97180 0.0001	0.90390 0.0001	0.94385 0.0001	0.93942 0.0001	0.94701 0.0001	0.66295 0.0001	0.96130 0.0001	0.69901 0.0001	1.00000 0.0	0.00643 0.9218
LBR	-0.06819 0.2979	-0.02990 0.6484	0.02353 0.7197	0.00753 0.9086	0.01824 0.7809	-0.01502 0.8189	0.02684 0.6823	0.04260 0.5158	0.02552 0.6971	0.02379 0.7168	-0.13903 0.0332	-0.33398 0.0001	0.01875 0.7749	0.02089 0.7500	0.00643 0.9218	1.00000 0.0

Table 7. The Factor Matrix For The Sixteen Measures of OP-C, OP-B, OP-D.

Measure	OP-B		OP-C		OP-D	
	Factor 1	Factor 2	Factor 1	Factor 2	Factor 1	Factor 2
1. n,	.78	-.17	.79	-.12	.78	-.17
2. n_2	.94	-.02	.94	-.02	.93	-.03
3. N_1	.97	.10	.98	,83	.97	.08
4, N_2	.97	.06	.97	.04	.96	-.05
5. N	.98	.09	.98	.07	.97	.07
6. N^A	.91	-.01	.96	-.00	.96	-.01
7. V	.96	.14	.97	.09	.96	.09
8. E	.89	.22	.90	.15	.88	.15
9. VG_1	.94	.09	.95	.08	.93	.10
10. VG_2	.77	.12	,95	.07	.93	.10
11. LOC	.94	-.25	.96	-.17	.95	-.19
12. B/C	.61	-.64	.7'2	-.50	.70	-.53
13. <;>	.97	.03	.97	.04	.97	.06
14. SP	.70	-.05	.60	-.01	.72	.04
15. NCLOC	.98	.05	.98	.05	.98	.05
16. LB-Ratio	-.03	.83	-.01	.92	-.02	.90
Percentage of explained variability	90	10	91	9	91	9

'J'able **8.** Summary **of Actual** Average Values **of** (VG_1) and **Values** Predicted **by Models** 1, 2, and **3.**

Model	l?c.lease	(V) Value Actual.. (A)	Predicted(P)	Delta (A)-(P)	Error(%) Delta ÷ (A)
1.	OP-B	4.45	5.00	-0.55	-12.40
	OP-C	4,53	5.09	-0.56	-12.40
	OP-D	5.30	5,70	-0.40	-7.50
Grand Average		4.76	5.26	-0.50	-10.60
2.	OP-B	4,45	4.59	-0.14	-3.10
	OP-C	4.53	4.86	-0.33	-7.30
	OP-D	5.30	5.27	-0.03	0.60
Grand Avc.rage		4.76	4.91	-0.15	-3,10
3.	OP-B	4,45	4.62	-0.17	-3.80
	OP-C	4.53	4.84	-0.31	-6.80
	OP-D	5.30	5.30	0.00	0.00
Grand Average		4,76	4.92	-0.16	-3.40

Table 9. Summary of Actual Average Values of (V) and Values Predicted by Models 4 and S.

Model	Release	(V) Value		Delta	Error(%)
		Actual (A)	Predicted(P)	(A)-(P)	Delta ÷ (A)
4.	OP-B	704	679	+25	+3.6
	OP-C	722	738	-16	-2,2
	OP-D	845	826	+19	-t-2.2
Grand Average		7 5 7	748	+9	+1.2
5.	OP-B	704	678	+26	+3.7
	OP-C	722	735	-13	-1.8
	01'-11	845	826	+19	+2.2
Grand Average		757	746	-10	+1.3

‘1’able JO. Characteristics of Four independent Segments of Software

Segment Number	Number of Modules	Actual Average VG_1	Value V	NCLOC
1.	16	16.4	3343	102
2.	16	17.9	4016	139
3.	so	8.16	1823	64
4.	55	11.10	2212	71

Table 11. Sample Calculation of **Actual Average Values** of **(VG1)** and **Values Predicted by** Model **1For** Segments 1 **to** 4.

		(V) Value		Delta	Error(%)
Model	**Segment**	**Actual(A)**	**Predicted(P)**	**(A)-(P)**	**Delta ÷ (A)**
1.	1	16.40	**18.19**	-1.79	**-10.9**
	2	**17.90**	21. S6	-3.66	-20.4
	3	S.16	10. **59**	-2.03	-24.4
	4	**11.10**	12. **54**	-1.44	-13.0
Grand Average		**13.39**	**15.72**	-2.33	-17.3

Table 12. Summary of Actual Grand Average Values of (VG1) and Values Predicted by Models 1, 2 and 3 For Segments 1 to 4.

Model	Segment	VG_1 Grand Average Value		Delta	Error(%)
		Actual(A)	Predicted(P)	(A)-(P)	Delta ÷ (A)
1.	1 to 4	13.39	ls. s7	-2.33	-17.3
2.	1 to 4	13.39	13.31	+0.08	+0.6
3.	l t o 4	13.39	13.48	-0.09	+0.7

Table 13. Summary of Actual Grand Average Values of (V) and Values Predicted by Models 4 and S For Segments 1 to 4.

		VG_1 Grand Average Value		Delta	Error(%)
Model	Segment	Actual(A)	Predicted(P)	(A)-(P)	Delta÷(A)
4.	l t o 4	2848	2570	+278	+9.7
5.	1 to 4	2848	2S71	+277	+9.7

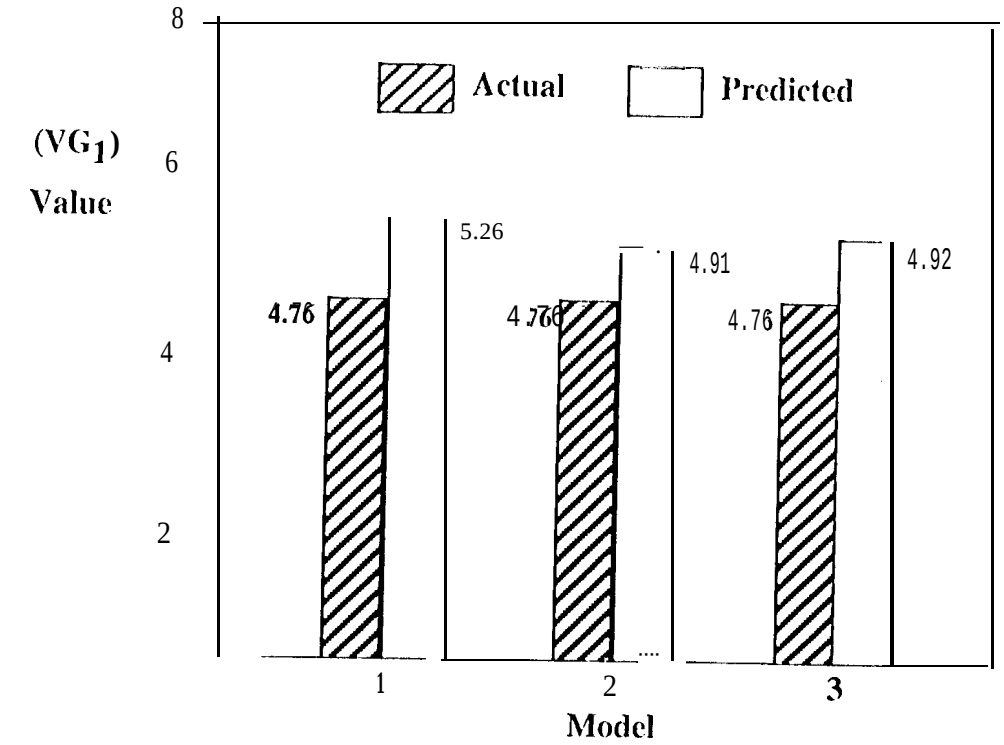

Figure 3. Actual Average Values of (VG $_1$) and Values Predicted by Models 1, 2 and 3.

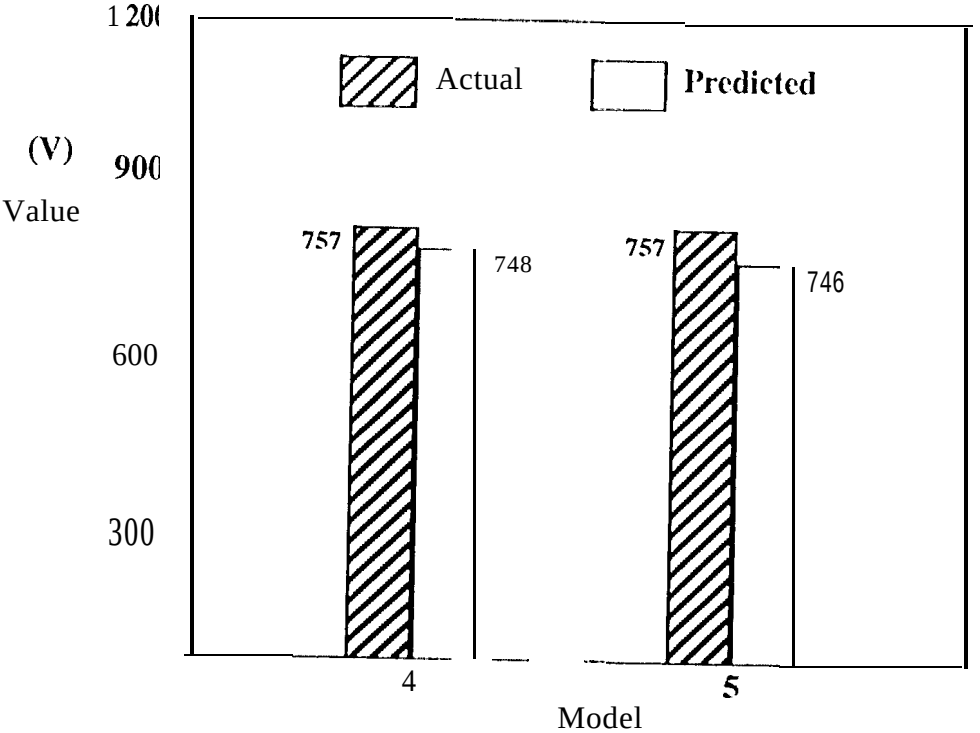

Figure 4. Actual Average Values of (V) and Values Predicted by Models 4 and s.

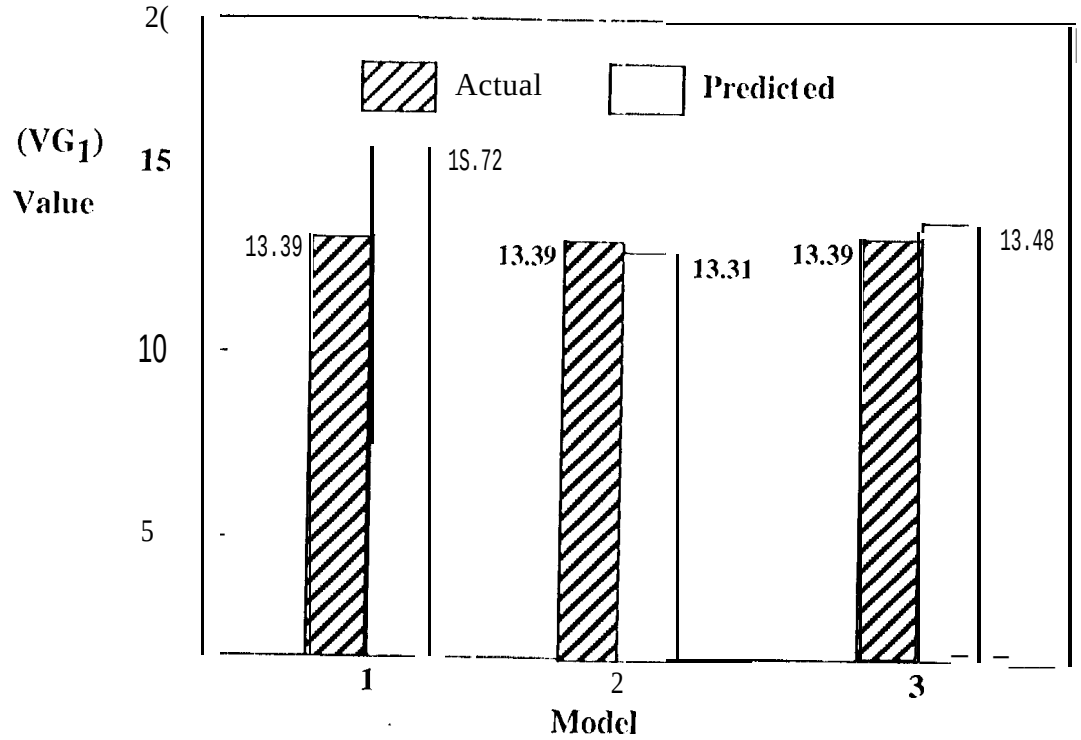

Figure S. Actual Average **Values** of (VG $_1$) and Values **Predicted by Models 1, 2** and **3** for **Independent** Segments of Software.

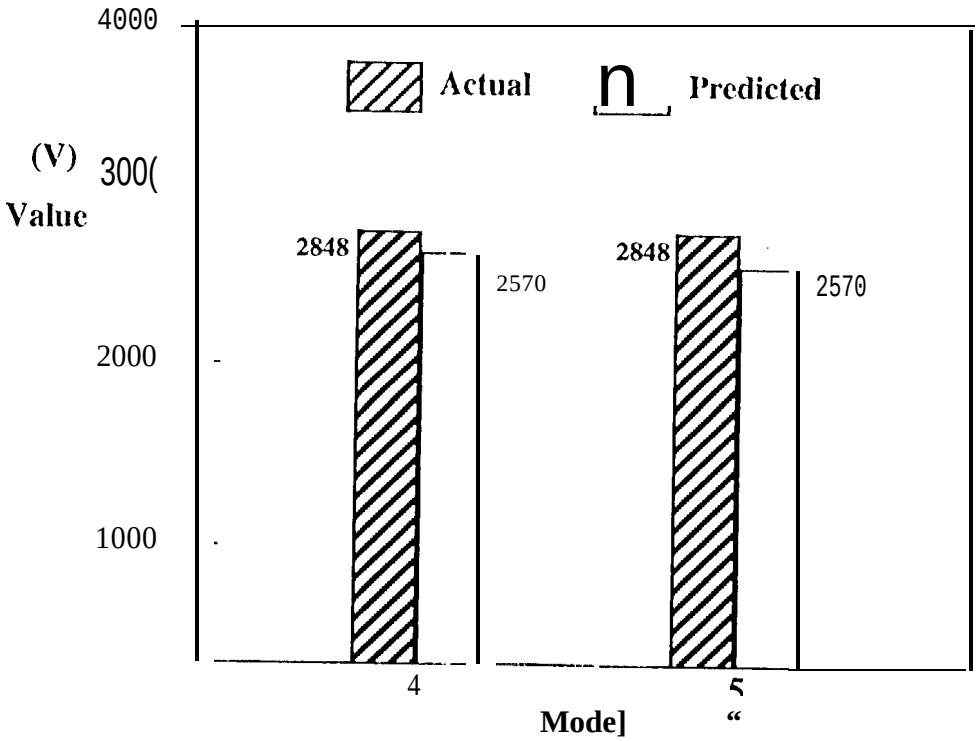

Figure 6. Actual Average **Values** of (V) and **Values Predicted by Models 4 and 5 for Independent Segments of Software.**