EXPOSING USEFUL TRENDS IN METRIC DATA THROUGH GROUP LEVEL ANALYSIS

by

Dr. Dennis Kafura
Dr. James Canning

TR-85-32

August 1985

Exposing Useful Trends in Metric Data Through Group Level Analysis

Dr. Dennis Kafura
Department of Computer Science
Virginia Polytechnic and State University
Blacksburg, Virginia 24060

Dr. James Canning
Department of Computer Science
University of Lowell
Lowell, Massachusetts

August 8, 1985

ABSTRACT

In this paper the results of experiments which applied both structure and code metrics to three large scale systems are presented. This metric research is distinct in that trends in the data are uncovered through the use of group level analysis. Components are partitioned into groups based on their various metric values and on observed measures of complexity (ie. errors, coding time). Crosstabulation data is given which indicates that trends between some of the metrics and the observed data do exist. Code metrics typically formed groups of increasing complexity which corresponded to increases in the mean values of the observed data. The strength of the Information Flow metric and the Invocation measure is their ability to form a group containing highly complex components which was found to be populated by outliers in the observed data.

PART I

INTRODUCTION

One of the basic objectives of software engineering is to transform the creation of software systems from an artistic, poorly understood, and even undisciplined, activity into a carefully controlled, methodical, and predictable enterprise. To make software an engineerable product, it is important that the designers, implementors and maintainers of software systems be able to express the characteristics of the system in objective and quantitative terms (e.g. software performance [Lync81], and software reliability [Musa75]) those characteristics relating to software quality are typically evaluated only in qualitative terms.

Quantitative measures of quality are collectively referred to as software metrics. One of the key --- and, unfortunately, too often neglected --- parts of software metric research is the validation of the metric on realistic software systems. The validation, however is vital because it establishes the relationships between the measures of software products (e.g., a complexity metric) and important resources in the software process (e.g., errors, coding time). One form of a validation analyzes the relationship between metric values and historical project data. Such validations provide the compelling evidence needed to gain acceptance and use of software metrics as a standard industry wide practice.

The purpose of this paper is to report on experiments which identified and compared groups of components. This group analysis is in contrast to the component by component correlations generated by most prior software metric studies[Kafu85] [Basi83]. The motivation for grouping components together is to identify overall trends in the data which may not be evident from simple linear or rank correlation techniques. Previous validations of software metrics have relied heavily on measures of linear or rank correlations to determine the degree of relationship between software metrics and features of the systems being analyzed. For example, Basili's careful and extensive validation[Basi83], and our own experience with data taken from the same source [Kafu85] [Cann85] shows inconsistent or, frankly, poor correlations between any metric and project resource data. Is this to suggest that the software metrics are of little or no value? In the same manner, should we be disturbed to find that one component with a high metric value has no errors while another component which has a low metric value has numerous errors? When linear and rank correlations are used the answer to these two questions would appear to be yes. However, past research [Henr81a] which grouped components, identified a strong correspondence between information flow based groups and their error properties. Furthermore, the discussion and the results which follow generally indicate that positive trends do in fact

exist between some software metrics and the developmental data.

There are three reasons for studying the behavior of a group of components. First, measurement of component groups does not require the metrics to differentiate between components having approximately the same complexity value. Small differences in complexity values and developmental data readings between components should not mask out any overall trends in the data. This masking is possible when using the non-parametric Spearman rank correlation technique as a measure of metric validation. For example, consider the hypothetical data given in table 1 for components A through J. Notice, that the rankings imposed on these ten components by the CODING TIME data and by the information flow metric, INFOFLOW, are quite different yielding a rank coefficient of -0.03. From this result, one may conclude that CODING TIME and the INFOFLOW metric are highly unrelated. However, a group analysis would determine that a strong relationship does exist between these two measures. The two orders of magnitude difference between the reported CODING TIME for components A through H and both component I and component J is reflected in their large difference in INFOFLOW values. The inability of the non-parametric Spearman Rank method to identify some of the relationships between the metrics and the data occurs because it does not consider the distances between the various components. How-

ever, parametric techniques which account for the distances between data points, such as Pearson's correlation method also offered little insight since the distribution of the data was typically asymetrical.

TABLE 1

Hypothetical Data

COMPONENT	CODING TIME	INFOFLOW
A	0.50	9.3 x 10**2
B	0.73	8.5 x 10**2
C	1.03	8.2 x 10**2
D	1.05	7.7 x 10**2
E	1.11	7.3 x 10**2
F	1.22	5.9 x 10**2
G	1.29	4.7 x 10**2
H	1.33	4.5 x 10**2
I	126.00	3.8 x 10**6
J	133.86	3.4 x 10**6

A second reason for studying groups is to better understand those components which are considered to be outliers. It may be unrealistic, and even unnecessary or inefficient, to assume that the development staff has the resources and motivation to give each component extra consideration. By identifying those components which have extremely high values, management can better assess how to distribute the workload for an project.

A third motivation for performing a group analysis is that it is less sensitive to anomalous data points. These anomalies occur either when a component has a high metric reading with a low developmental data value or when a component has a low metric reading with a high developmental data value. The existance of these anomalies often leads to linear or rank correlations which indicate weak associations between the metrics and the developmental data. Given the state of the art in both software metrics development and data collection systems, it is not surprising that these anomalies exist. Those components which possess a high metric reading, but have little or no reported errors, may have been more thoroughly tested before being submitted for configuration management. It is equally possible that despite employing data filtering techniques, the developmental data for a few components may not have been accurately reported. By analyzing the components according to groups, these anomalies can be better identified. Furthermore, their existance are less likely to inhibit the identification of any trends which may exist in the data.

The remainder of this paper contains five sections organized as follows. The following section provides a brief description of the ten software metrics used in this research along with the software that was analyzed. The next section, entitled GROUPING METHODS, describes three techniques which were used to establish component groups. More than one

grouping technique was initially explored to investigate the sensitivity of the results to a particular grouping method. The third section entitled, DEVELOPMENTAL DATA MEANS WITHIN GROUPS, contains a subset of the results which indicated that group level analysis show trends in the data. In particular, mean values for ERRORS, WEIGHTED CHANGES and CODING TIME were generally higher for groups containing components of higher complexity. The section entitled, DEVELOPMENTAL DATA GROUPS CROSSED WITH METRIC GROUPS, presents additional evidence that some software metrics are related to both count based and time based measures. In this fourth section, data from a collection of two-way crosstabulation tables is displayed. The fifth and last section provides a brief summary of the major points made in this paper.

PART II

METRICS AND RESOURCES USED IN THIS RESEARCH

The quality of a software product is determined by numerous quality attributes which emerge in increasingly detailed form as the life cycle progresses. Accordingly, numerous quantitative measures must be employed throughout the life cycle -- each measure defined to quantify one of the many quality attributes. Three general classes of software metrics can be distinguished: measures based on an automated analysis of the system's design structure, termed "structure metrics", measures based only on implementation details, termed "code metrics" and measures which are a combination of the other two, termed "hybrid metrics". The use of all three types of metrics is important for two reasons. First, these classes of metrics appear to be measuring different aspects of the system quality. In the original Information Flow study, Henry et al.[Henr81b] found their structure metric to be orthogonal to three code metrics. Similarly, Kafura et al.[Kafu84], presented additional evidence that structure metrics and code metrics were indeed measuring different dimensions of software complexity. Fundamentally, the quality of a system is too multifaceted to be measured by any one metric or by metrics in only one of these classes. Second, each class of metrics can be first used at different points in the software life cycle. While code and hybrid metrics may be useful indicators during the testing

and maintenance phases, they come too late in the software life cycle to address fundamental design decisions. Structure metrics, on the other hand, can be taken early in the life cycle since they are based only on system features which emerge at the high-level design phase.

There are three code metrics used in this study. The first code metric is a simple size measure: lines-of-code (LOC). The second code metric is Halstead's software science "effort" measure[Hals77]. The third code metric is the measure of cyclomatic complexity defined by McCabe[McCa76]. Several carefully designed experiments have shown that meaningful relationships exist between these metrics and significant software characteristics (for example, [Curt79]). Properly used, these metrics can play a useful role during the later phases of the software life cycle (testing, acceptance, maintenance, etc.). However, serious objections have been raised against the experimental design and statitistical treatment of a number of the software science experiments [Hame82][Lass81][Shen83]. Even if these objections can be satisfied, code metrics are available too late in the life cycle to exert a major impact on the life cycle characteristics and thus, should not be used alone.

As indicated above, a structure metric is defined in terms of some observed connection between components of the system which have emerged during the (high level) design phase. Many design methodologies, typified by Ross' SADT

[Ross77] and Yourdan and Constantine's SA-SD [Your79], focus on identifying the components in the system and specifying how these components are structurally related through control and/or data connections. Accordingly, structure metrics use only these features, components and relationships among components, to define a numerical measure. In total, these connections define the system structure. Each metric focuses on some aspect of structure which affects complexity, comprehensibilty, maintainablilty, etc.. The four structure metrics surveyed below have been incorporated into our work.

A structure measure based on the data relationships among components is the imformation flow metric mentioned above [Henr79]. This metric identifies the sources (fan-in) and destinations (fan-out) of all data related to a given component. The data transmission may be through global data structures, parameters, or side-effects. The fan-in and fan-out are then used to compute a worst-case estimate of the communication "complexity" of this component. This complexity measure attempts to gauge the strength of the components's communication relationships with other components.

Another structure metric is McClure's metric [McCl78] which attempts to measure "invocation complexity". The relevant features in analyzing the invocation complexity of a component are: (1) all variables which control (via conditional or iteration statements) the invocation of the component; and (2) all components which can affect the value of

these variables. In this metric a small invocation complexity is assigned to a component which, for example, is invoked unconditionally by only one other component. A higher complexity is assigned to a component which is invoked conditionally and where the variables in the condition are modified by remote ancestors or descendents of the component.

A third structure metric, defined by Woodfield [Wood80], is based on the concept of "review complexity". Woodfield observes that a given component must be understood in each context where it is called by another component or affects a value used in another component. In each new context the given component must be reviewed. Due to the learning from previous reviews, each review takes less effort than the previous ones. Accordingly, a decreasing function is used to weight the complexity of each review. The total of all of these weights is the measure assigned to the component. Woodfield applied this measure successfully in a study of multi-procedure student programs.

The fourth and final structure metric used in out research is the "stabilty" measure defined by Yau and Collofello[Yau 80]. This measure directs attention to the ripple effect which occurs when a change to one component necessitates changes to other components. In this measure the flow of data through parameters and global variables is used to identify all possible components which could be affected by a change to a given component. Finally, three hybrid me-

trics are used. Each of these hybrid measures is a modification of one of the structure metrics. A hybrid metric determines a measure for a component by weighting the structure measure for that component by a code measure. To simplify matters, we have used only lines-of-code as the weighing term. Using any of the other code metrics does not change the overall character of the results. It should be noted that three of the ten metrics used in this project (Henry and Kafura's information flow, Woodfield's review complexity, and Yau and Collofello's stability measure) were originally posed by their authors as hybrid metrics. The three hybrid metrics are: information flow weighted by lines-of-code, Yau and Collofello's stability weighted by lines-of code, and Woodfield's metric weighted by lines-of-code. The McClure invocation complexity metric does not lend itself to such weighting.

For reference, the ten metrics described above will be referred to in subsequent tables and discussion by the following abbreviations:

1. LOC (lines-of-code)
2. EFFORT (Halstead's software science effort)
3. INFOFLOW (Henry and Kafura's unweighted information flow complexity)
4. INVOKE (McClure's invocation complexity)
5. REVIEW (Woodfield's review complexity)
6. STABILITY (Yau and Collofello's stability measure)

7. INFO-LOC (weighted information flow)

8. REVIEW-LOC (weighted review complexity)

9. STAB-LOC (weighted stability measure)

Also, recall that the first three of the metrics are code metrics, the middle four are structure metrics, and the last three are hybrid metrics.

Of the four structure metrics only the Information flow metric has been previous applied and validated on a realistic non-trivial program -- the UNIX operating system[Henr79][Henr81][Henr84]. Two of the structure metrics, Woodfield's Syntactic Interconnection Model and McClure's Invocation complexity were applied with success on smaller scale programs. Woodfield[Wood80] conducted carefully controlled experiments which measured programs built by students competing in a programming contest and correlated these measures with coding time. McClure[McCl78] applied the invocation measure to a self-built COBOL program and provided a subjective evaluation. The fourth structure metric used in this research, the Stability measure of Yau and Collofello[Yau 80] was proposed in their original work, but no validation was performed by the authors.

This research has added to the knowledge base with regard to these four promising structure based metrics. The previous work of Kafura and Henry has been augmented by applying their Information Flow metric to software possessing radically different characteristics than the Unix operation sys-

tem. The current study measured three commercially developed scientific programs each approximately 100,000 lines of FORTRAN source code. The initial work of Woodfield and of McClure has been extended by exploring the usefulness of their measurement on non-trivial programs. Furthermore, the research of Yau and Collofello has also been extended since their promising measure has been automated and experimented with on production software.

Critical resources necessary for this project were provided by the Software Engineering Laboratory (SEL) headed by Frank McGarry, Jerry Page and Dr. Victor Basili. This organization, formed in 1976, is composed of three members: Nasa/Goddard Space Flight Center(GSFC), The University of Maryland (Computer Science Department), and Computer Science Corporation (Flight Systems Operation). The SEL has defined and implemented an extensive monitoring and data collection process by which the details of all aspects of the software development process and product could be extracted for analysis [Nasa 81].

One critical resource provided by SEL management was the source code for three large scale projects, each of which was written in FORTRAN. These three software systems contain approximately 100,000 lines of source and can be categorized as scientific in nature. Each project is a collection of FORTRAN subroutines and functions referred to in this paper as components.

In addition to the three FORTRAN systems, SEL also provided a developmental database associated with each of the three projects. The database itself contains numerous files and a wide range of information. Selected for this study were the following variables:

1. Component Changes: A modification to a component made either to correct an error, to improve system performance, to add capability, or to implement a requirements change.
2. Component Errors: A discrepency between a specification and its implementation. The specification might be requirements, design specification, or coding specification.
3. Coding Time: The time recorded by SEL personnel just to implement an individual component.

In addition to these variables, a variable termed, WEIGHTED CHANGES, was derived for the purposes of this research. This is a measure of the total amount of effort spent either to fix an error or to make a change to a given component[Basi83]. In the Nasa/Goddard environment, programmers classify each change by estimating the amount of effort needed to isolate the change/error and to implement the change/error. The four possible classifications recorded in the developmental database are:

1. Less than one hour.
2. One hour to one day.

3. One day to three days.

4. Over three days.

Corresponding to each of these classifications are the four weights suggested by Basili: 0.5, 4.5, 16.0, or 32.0 hours. Thus, a component's WEIGHTED CHANGE value is derived by summing the hours attributed to the component for every change with which it was involved.

Because of the nature of the data collection process and the enviroment in which the systems were constructed, not all of the components from the three projects were used in this experimentation. The description of two subsets of the Nasa/Goddard data that were used in the following analyses concludes this section. One subset of components, referred to as DATASET A, was utilized in those experiments which compared the metrics with the count based dependent variables (ERRORS, CHANGES, WEIGHTED CHANGES). In particular, DATASET A does not include those components which were entirely or largely reused from prior projects. The second subset of components, known as DATASET B, was used in those experiments which related the metrics to time base dependent variables(CODING TIME). This set of components is a subset of DATASET A. The selection process, or "filter", used to produce DATASET B, was found to be necessary in order to adjust for problems with the data reporting mechanisms. A more detailed description of each of these two sets of components is presented below.

One subset of components (DATASET A) used for this research contains components which will be used when analyzing the relationship between the metrics and the count based measures. DATASET A is a collection of those subroutines and functions which were either totally new components or extremely modified old components. The SEL database classifies each component as either: new code, extremely modified old code, slightly modified old code, or an exact copy of old code. It is necessary to omit slightly modified or duplicated code from consideration since error counts and development times for reused components do not accumulate from project to project in the SEL reporting mechanism. These reused components, having already undergone rigorous testing and debugging when intially implemented, would distort the count based measurements. Similarly, the time based measurements would also be biased since they would not reflect the original effort expended in the development of these slightly modified or duplicated components. It was decided that extremely modified components should be incorporated into the analyses since these components undergo a significant transformation. It was felt that count based measures and time based measures would approximate the values of such a component had it been completely constructed anew. In support of these decisions, table 2 below provides the mean values for the four measures across the four component classifications. An analysis of variance was performed using

dependent variables (ERRORS, CHANGES, WEIGHTED CHANGES, TOTAL TIME) to test the null hypothesis that mean values for the four component classes were not statistically different. In each of the four analyses of variance the null hypothesis was rejected (alpha=0.05). Furthermore, Fisher's Protected Least Significance Test (LSD) was used to identify those means which were significantly different. In all four tests, it was found that means from component class (1) and component class (2) were not significantly different, but the means for these two classes were significantly different from both class (3) and class (4).

TABLE 2

Data Means for Various Component Types

Component Class	Errors	Changes	Weighted Changes	Total Time
(1) New Code Components	2.03	2.11	44.20	16.09
(2) Extremely Modified Components	1.47	2.62	43.57	13.32
(3) Slightly Modified Components	0.43	1.11	18.98	6.06
(4) Duplicated Components	0.17	0.52	3.24	3.37

Another subset of the Nasa/Goddard components(DATASET B) was used for experiments involving time based dependent variables (Coding Time). This collection of components not only omits slightly modified routines and duplicated code from the experimentation, but also eliminates components which fail to pass two additional criteria. Both of these criterion provide a validity check on the reporting of time based dependent variables.

The first criteria eliminates those components which were constructed by programmers who were identified as poor or inconsistant reporters of time based dependent variables. Programmers were considered poor reporters if the ratio, Vm, found below, was less than eighty percent. This ratio, first suggested by Basili et al. [Bas83], utilizes the partial redundancy built into the SEL monitoring process. Each week every programmer files a Component Status Report (CSR) describing the time they spent on each module. Also on a weekly basis, the project manager files a Resource Summary Form (RSF) recording the time each programmer spent on the project during that week. Basili et al. have indicated that the manager reported information found in the Resource Summary Form is considered to be the more accurate. Thus, if a given programmer fails to submit a Component Status Report for a given week, this would lower his Vm ratio. The Vm ratio is defined as:

Number of weekly CSR's submitted

by programmer

$$Vm = \frac{\text{Number of weeks programmer appears on RSF's}}{\text{Number of weeks programmer appears on RSF's}}$$

The second criterion is based on the fact that time based dependent variables are not only reported for individual components, but are also reported for individual subsystems. Programmers typically report their time spent on an individual component basis. However, a programmer may choose to attribute work hours to an entire subsystem if these hours cannot be accurately partitioned among that subsystem's individual components. When the time reporting is done on a subsystem level, information at the component level is lost. If less than eighty percent of a subsystem's reported time is attributed to individual components, then these components are not incorporated into DATASET B.

PART III

GROUPING METHODS

The initial group analysis utilized three grouping techniques. It was important to investigate the effect of different grouping methods to gain confidence that overall results were not sensitive to a particular grouping technique. One grouping method, referred to as the Logged Data technique, identifies six distinct groups. Two of the groups consist of those components with values lying within one standard deviation from the mean (plus or minus). A second pair of groups contain components with values lying within two standard deviations of the mean(plus or minus). Finally, those components lying more than two standard deviations from the mean comprise the last two groups. However, since most of the original data is asymmetric with positive skew the usefulness of univariate summary statistics, such as the mean and the standard deviation is minimal. Meaningful summary statistics can be obtained, however, for asymmetric data, by first (a) transforming the data to acheive a symmetric distribution, (b) calculating the mean m, and standard deviation s of the transformed data, and then (c) applying the inverse transform to the values: m-2s, m-s, m, m+s, m+2s to determine the group boundaries of the original data. This method of transforming the data to obtain a non-skewed distribution has been used by Crawford et al.[Craw85] to analyze static metrics. In their original

work, as is done in this research, a logarithmic transformation is applied to the basic data.

The second grouping technique, referred to as Clustering, is based on a hierarchical clustering scheme proposed by Johnson[John67]. Basically, the algorithm begins by forming one cluster for each observation in the data. Components are then grouped according to a distance function, clustering together those components which are "neighbors". Clusters are determined by considering the distances between components within a given cluster versus the distances between the clusters. Ideally, the algorithm will choose the optimal number of clusters, but this research will arbitrarily fix the number of clusters to be six. Fixing the number of clusters constant at six, facilitates any comparisons made between this grouping technique and the other two techniques.

The third grouping technique, referred to as Rank Grouping, is the simplest of the three. Once again the number of groups was held fixed at six for ease of comparison. This method defines the groups according to the ranking of the components imposed by a given metric. The set of components are first ranked in ascending order by a given metric and then placed into groups according to rank. Components whose rank was less than one-sixth the total number of components were assigned to group 1, components whose rank was greater than one-sixth the total number of components but less than

two-sixths the total number of components were assigned to group 2 and so on. Whenever two or more components had the same complexity measurement their ranks were redefined as the mean of their corresponding ranks. Since it is possible that numerous components may have the same complexity value, and in some instances this is likely(ie. REVIEW Metric) not all the groups will necessarily contain the same number of members. In the worst case, a rank order group may contain no components when this simple rule is used.

PART IV

TRENDS IN DEVELOPMENTAL DATA MEANS FOR EACH METRIC GROUP

The purpose of this section is to present results of initial experiments which grouped components together. These group level experiments are classified by the following three factors:

1. Grouping Technique
2. Complexity Metric
3. Developmental Data

Since there are three distinct grouping techniques, ten different complexity metrics and three types of developmental data used in this research, a total of ninety results were generated and analyzed. This large amount of information has been pruned and summarized in the following five tables. These tables are representative of the total ninety cases. The first three tables contain the mean number of ERRORS for each of the three grouping techniques. By comparing these three tables any difference in the grouping techniques can be seen. The last two tables contain the mean number of CODING HOURS and the mean number of WEIGHTED CHANGES for only the Logged Data grouping method. These two tables were included so as to present results which spanned across all three of the developmental data variables utilized in this experiment. The rows of each of the five tables indicate which metric was used to identify group boun-

daries, while the columns represent the six different groups, numbered from 1 (least complex components) to 6 (most complex components).

TABLE 3

Mean Errors Per Group for Logged Data Technique

	GROUP NUMBER					
	1	2	3	4	5	6
LOC	0.10	0.65	1.12	2.93	4.10	4.28
CYCLO	0.23	1.21	1.09	3.29	3.46	3.12
EFFORT	0.67	0.78	1.57	2.95	2.80	2.55
INFOFLOW	1.07	1.62	1.49	1.98	3.58	23.50
STABILITY	1.25	1.51	1.90	2.62	2.50	---
REVIEW	5.05	---	1.37	2.24	2.80	---
INVOKE	1.25	0.67	1.17	2.19	5.85	20.75
STAB-LOC	1.08	1.53	1.57	2.48	4.91	0.20
REVIEW-LOC	5.34	0.48	0.98	2.28	4.08	3.55
INFO-LOC	0.43	1.50	1.34	2.03	3.91	17.00

Table 3 contains the mean number of ERRORS for groups generated by the the Logged data grouping technique. The results presented in this table largely indicate that those groups containing components with higher complexity values have a higher mean number of ERRORS associated with them.

An example of this trend can be seen by observing the LOC metric. Note the steady increase in the mean number of ERRORS across the six groups. The other two code metrics, McCabe's CYCLO measure and Halstead's EFFORT measure also show a similar tendency. Of the four structure metrics, only Woodfield's REVIEW metric fails to exhibit this behavior. It is striking to note the dramatic jump in the mean number of errors between Group 5 and Group 6 for both the INFOFLOW metric(from 3.58 to 23.5) and McClure's INVOKE complexity measure(from 5.85 to 20.75). Of the three hybrid metrics, the INFO-LOC measure and the STAB-LOC measure both impose groupings on the components which relate to the average number of errors reported. The trend is consistent for the first five groups established by the STAB-LOC metric, while the sixth group exhibits a decrease in the mean number of errors. The sharp increase between group 5 and group 6 is also present for the INFO-LOC groups, jumping from 3.91 mean errors to 17.0 mean errors. The REVIEW-LOC metric fails to show any strong relationship between the mean number of errors and the reported complexity. In fact, these results give conterintuitive evidence that components which have the smallest REVIEW-LOC complexity will, on the average contains more errors that any other group. Three entries in the table contain dashes implying that the Logged Data grouping technique defined group boundaries in such a manner that these groups were empty.

Table 4 contains the mean number of ERRORS per group for each of the ten software metrics when the Clustering technique is employed.

TABLE 4

Mean Errors Per Group for Clustering Technique

	GROUP NUMBER					
	1	2	3	4	5	6
LOC	0.88	2.49	4.29	4.90	4.00	4.66
CYCLO	0.98	3.03	3.50	4.23	2.60	2.66
EFFORT	2.12	2.57	2.64	0.50	4.00	2.00
INFOFLOW	1.84	4.64	3.77	3.00	23.00	44.00
STABILITY	1.39	1.97	2.23	3.20	2.00	---
REVIEW	6.07	1.09	1.37	2.42	3.33	1.77
INVOKE	1.03	2.27	4.72	6.93	22.00	13.00
STAB-LOC	1.44	2.65	2.76	5.00	5.91	0.25
REVIEW-LOC	1.60	2.44	3.53	4.70	4.00	5.00
INFO-LOC	1.87	4.40	4.00	2.00	17.00	44.00

The results in this table are for the most part, similar to those found in the previous table when the Logged Data technique was employed. Once again a large increase in the mean number of ERRORS exists between adjacent groups for the INFOFLOW metric, the INFO-LOC metric, an the INVOKE metric.

For both information flow based metrics this large increase exists between groups 4 and 5 as well as groups 5 and 6. For McClure's metric, the mean number of errors for group 5 (22.00) and group 6 (13.00) are noticeably higher than the means for the four lower groups. Furthermore, the mean number of errors across clustered groups for the LOC metric, the CYCLO metric, the STABILITY metric, the REVIEW metric, and the STAB-LOC metric all exhibit behaviors which are similar to their Logged Data counterparts. However, the behavior of the mean number of errors across clustered groups was somewhat different than Logged Data formed groups for two metrics. Halstead's EFFORT metric generates a decrease in the mean number of errors between clustered group 3 and clustered group 4. Also, Woodfield's REVIEW-LOC metric formed clustered groups which does in fact agree with intuition.

The data presented in table 5 contains the mean number of ERRORS found for the six groups when the Rank grouping technique was used. Although not as pronounced, the data still indicates that components belonging to higher complexity groups will tend to have, on the average, a larger number of reported errors. It is encouraging to find that even a naive grouping technique is sufficient to expose this general trend. For example, the ERROR mean for group 1 components for each of the three code metric groups(0.52, 0.84, 0.66) fall below one error, while the corresponding means

for the group 6 components (4.03, 4.01, 4.76) all exceed four errors. Furthermore, the error means for the four intermediate groups are consistently increasing. This trend is not as consistent for either the structure metrics or the hybrid metrics. Once again Woodfield's REVIEW and REVIEW-LOC metrics do not exhibit any consistent trend. However, both information flow based measures and McClure's INVOKE metric possess noticeably higher mean errors for group five and group six components than they do for component belonging to the lower four groups.

TABLE 5

Mean Errors Per Group for Ranked Data Technique

	GROUP NUMBER					
	1	2	3	4	5	6
LOC	0.52	0.75	1.73	2.15	3.85	4.03
CYCLO	0.84	1.08	1.25	2.48	3.24	4.01
EFFORT	0.67	0.67	1.18	2.71	3.06	4.76
INFOFLOW	1.46	1.35	1.59	1.64	2.50	4.50
STABILITY	1.45	1.41	2.43	1.68	3.13	2.95
REVIEW	5.05	---	1.37	---	2.24	---
INVOKE	1.25	0.82	1.03	1.50	2.51	5.95
STAB-LOC	1.03	1.52	1.58	2.24	2.62	4.03
REVIEW-LOC	3.09	0.55	1.25	1.94	2.21	4.03
INFO-LOC	1.35	1.04	1.87	1.67	2.27	4.88

As mentioned above, the following two tables contain results obtained from forming groups using only the Logged Data Technique. Results for the Clustering and Ranking methods were similar and are not presented. Table 6, found below, contains the mean number of WEIGHTED CHANGES for each of the groups established by the Logged Data Technique.

TABLE 6

Mean Weights Per Group for Logged Data Technique

	GROUP NUMBER					
	1	2	3	4	5	6
LOC	0.10	21.63	33.64	50.10	75.18	55.92
CYCLO	8.31	29.47	26.81	60.48	71.75	43.35
EFFORT	3.50	21.07	42.68	52.88	55.25	36.25
INFOFLOW	33.33	26.54	33.99	40.15	65.96	515.16
STABILITY	24.81	29.66	39.00	52.60	53.75	---
REVIEW	43.95	---	39.40	41.86	57.26	---
INVOKE	23.57	25.81	24.13	49.02	122.24	759.50
STAB-LOC	19.00	31.18	33.15	49.45	109.68	13.10
REVIEW-LOC	0.16	19.85	34.17	48.85	73.79	64.66
INFO-LOC	30.61	26.28	33.11	39.97	68.94	338.10

These numbers, with the exception of a few values, generally show that some metrics do place components into groups

which will have increasing weighted change means. Only Woodfield's REVIEW metric fails to indicate any relationship with WEIGHTED CHANGES. The three code metrics, together with the STAB-LOC and REVIEW-LOC each report a decrease in mean number of WEIGHTED CHANGES between group 5 and group 6. This decrease exists because these metrics positioned the single component with the highest WEIGHTED CHANGE value into group 5. McClure's INVOKE measure and both the INFOFLOW and INFO-LOC measures correctly placed this component into group 6, yielding the expected increase in the mean WEIGHTED CHANGES between group 5 and group 6.

The results given in table 7 illustrate the relationship between the mean number of CODING TIME hours and the six groups. The group were established using the Logged Data Technique. Although not everywhere consistent, once again, the pattern exists where those components belonging to higher groups require, on the average, a longer period of time to implement. Two of the three code metrics, LOC and Halstead's EFFORT, form groups which have ascending CODING TIME means. Three structure metrics, INFOFLOW, STABILTY, and INVOKE complexities each exhibit the same consistent trend. It appears that Woodfield's REVIEW metric is unable to impose a grouping upon the components that will relate to CODING TIME means. Of the three hybrid metrics, both the INFO-LOC metric and the STAB-LOC metric exhibit increasing CODING TIME means across the different groups. The

REVIEW-LOC metric also generates increasing CODING TIME means for groups two through six. However, this metric again attributes its least complex group of components with a relatively high average coding time (12.91).

TABLE 7

Mean CODING TIME Per Group for Logged Data Technique

	GROUP NUMBER					
	1	2	3	4	5	6
LOC	2.33	3.10	3.73	8.38	16.95	34.26
CYCLO	1.48	3.37	3.99	9.10	18.81	13.26
EFFORT	3.30	3.48	5.20	8.67	17.06	4.25
INFOFLOW	3.79	4.44	8.76	14.17	14.42	---
STABILITY	5.53	4.12	4.66	9.97	11.55	---
REVIEW	13.85	---	6.18	8.61	8.56	4.51
INVOKE	6.33	5.21	8.08	7.55	11.37	26.66
STAB-LOC	5.08	5.80	7.53	8.90	10.53	---
REVIEW-LOC	12.91	3.67	3.99	6.56	16.12	24.84
INFO-LOC	0.50	3.85	4.05	8.87	14.93	14.42

In order to statistically support the hypothesis that the ten software metrics impose a grouping on the components which yield significant differences in the three developmental data means(ERRORS,CODING TIME,WEIGHTED CHANGES), thirty

analyses of variance were performed. The results of these ANOVA's are summarized in table 8 found below. The results contained within this table indicate that twenty-five of the thirty analyses of variance rejected the null hypothesis that the group means were all equal(alpha=.05). The five anova's which failed to reject the null hypothesis are each denoted in the table with an asterisk.

TABLE 8

ANOVA Results Between Metrics and Developmental Data

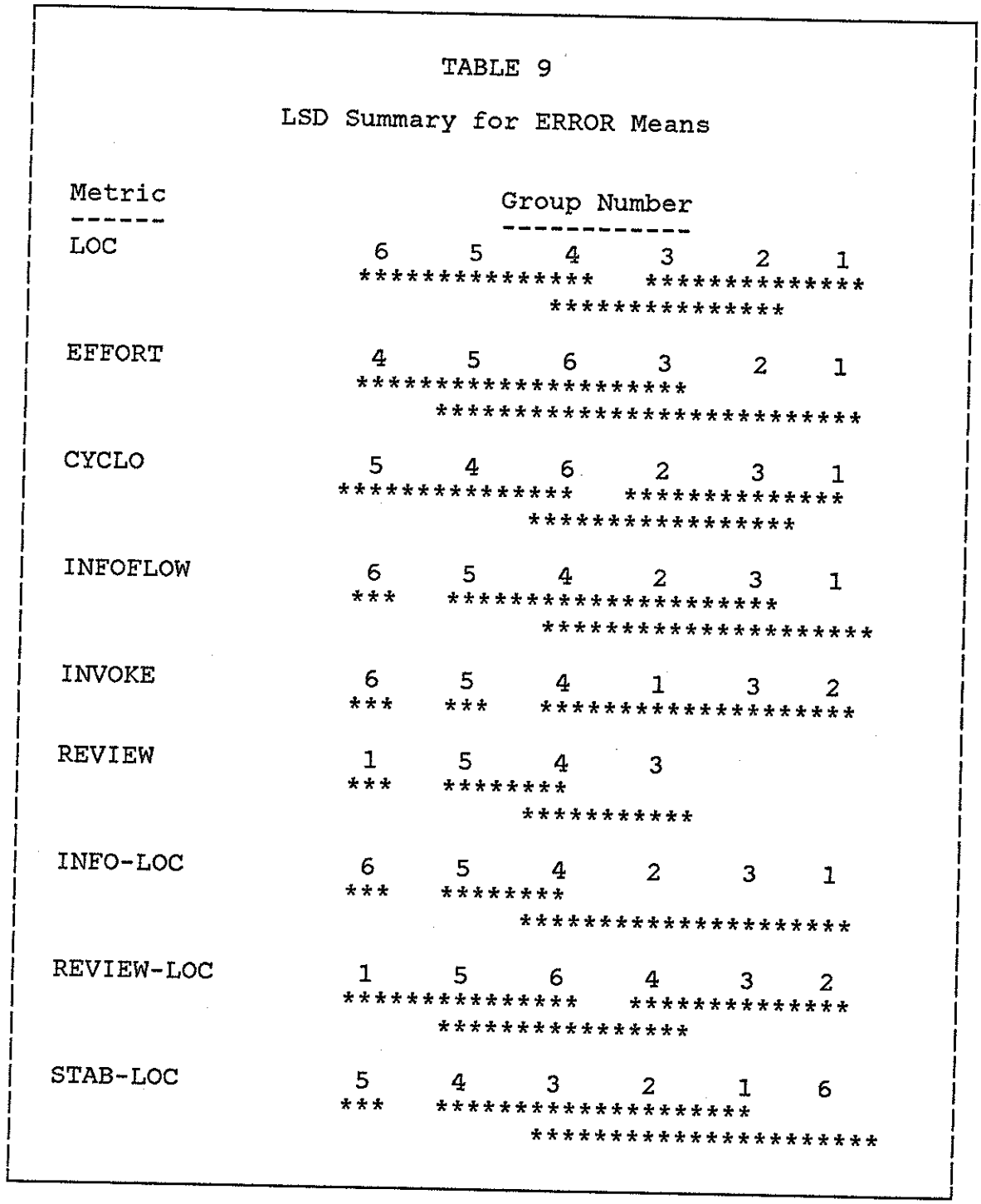
	ERRORS	WEIGHTED CHANGES	CODING TIME
LOC	REJECT	REJECT	REJECT
CYCLO	REJECT	REJECT	REJECT
EFFORT	REJECT	*	REJECT
INFOFLOW	REJECT	REJECT	REJECT
INVOKE	REJECT	REJECT	REJECT
REVIEW	REJECT	*	REJECT
STABILITY	*	*	REJECT
INFO-LOC	REJECT	REJECT	REJECT
REVIEW-LOC	REJECT	REJECT	REJECT
STAB-LOC	REJECT	REJECT	*

Although rejection of the null hypothesis statistically indicates that not all of the population means are equal, it does not indicate which group means are statistically different from each other. In order to identify those groups with statistically different means, a multiple comparisons procedure was applied to the data. In particular, Fisher's Protected Least Significant Difference (LSD) [OTT77], method was chosen since it is one of the few multiple comparison procedure's which allow for unequal group sizes. Table 9, table 10, and table 11 found below summarize the results of the Protected LSD multiple comparisons analysis used to determine which groups had significant differences in their ERROR means, their CODING TIME means and their WEIGHTED CHANGES means.

Within each table, group numbers are ordered, from left to right, by descending developmental data means. Those groups not underlined by a common line are declared to be significantly different according to the least significant criterion. For example, in table 9 McClure's INVOKE metric defines the mean number of ERRORS for Group 6 to be significantly different than the mean number of ERRORS for the other five groups. Furthermore, the ERROR mean for Group 5 was also considered different than the ERROR means for all other groups. However, the ERROR means for Group 4, Group 1, Group 3 and Group 2 were not found to be statistically different. In a very similar manner, the INFO-LOC metric

TABLE 9

LSD Summary for ERROR Means



declares Group 6 ERROR means to be statistically different from the remaining five groups. Although the Group 5

ERROR mean is not statistically different from the Group 4 ERROR mean, it is declared to be different from the means of the lower three groups.

Three observations can be made regarding the data found in these three tables. First, four metrics: LOC, INVOKE, INFOFLOW and INFOFLOW-LOC form more meaningful group boundaries than do the other six metrics. These metrics will generally place those components with higher development data values into the more complex groups. This is unlike the REVIEW based metrics which typically assigns components with a high mean value to its least complex group. Second, the LOC, INVOKE, INFOFLOW and INFO-LOC metrics are not able to identify significant differences between the means of the least complex groups. Third, the strength of McClure's metric and to a great extent the two information flow metrics are their ability to identify groups of highly complex components which have mean values significantly higher than the mean values for the less complex groups.

TABLE 10

LSD Summary For CODING TIME means

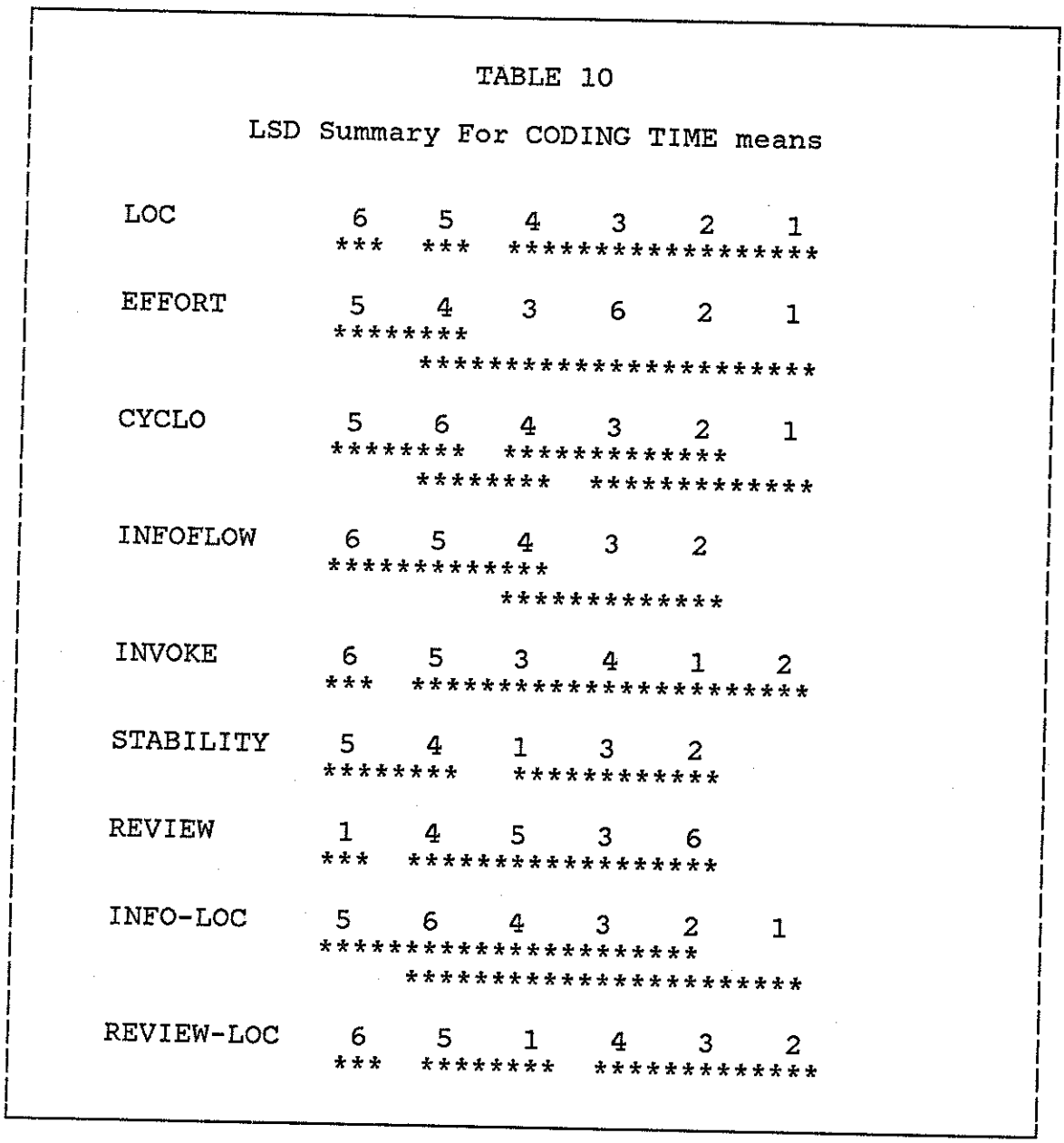
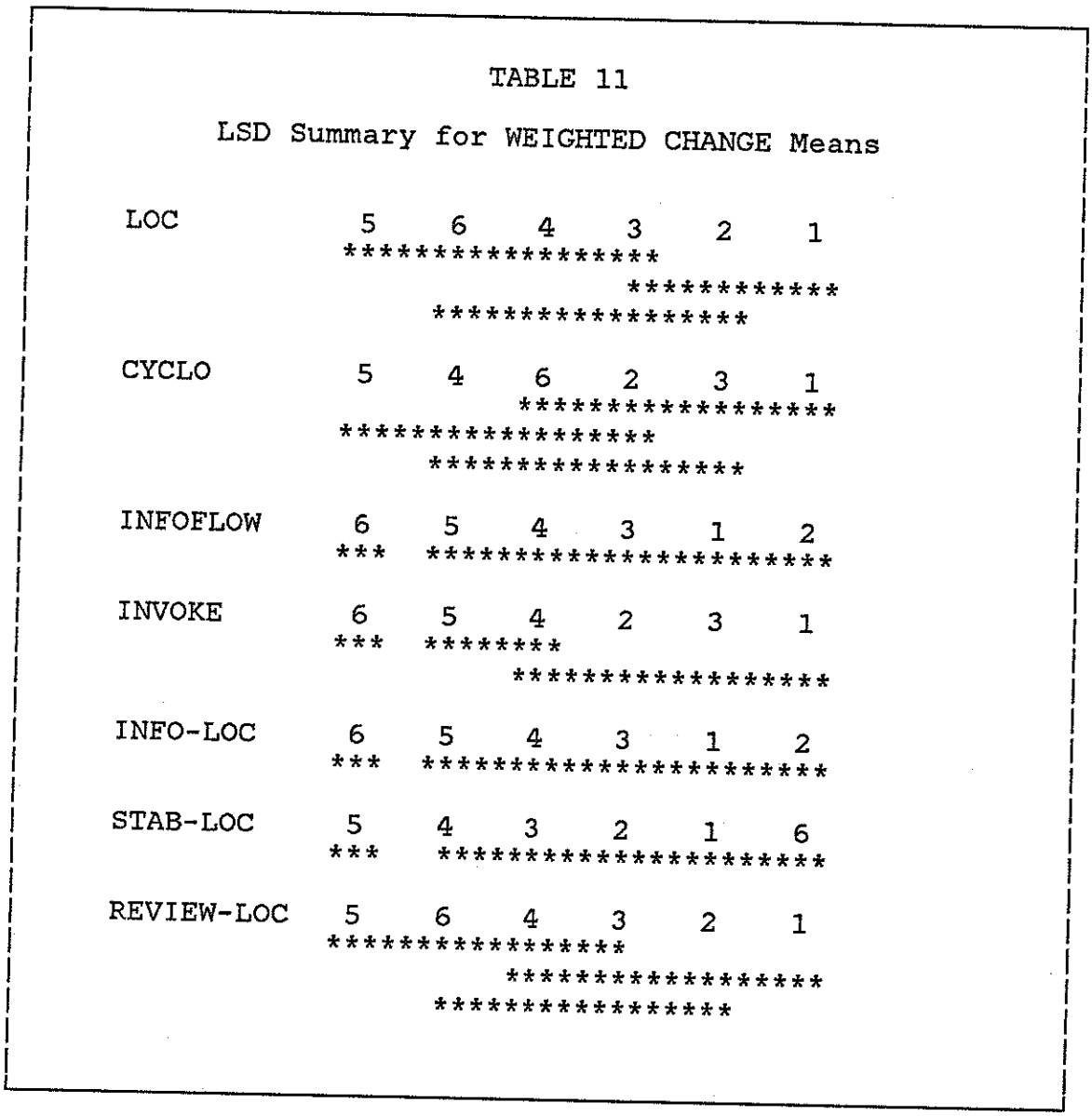


TABLE 11

LSD Summary for WEIGHTED CHANGE Means



PART V

METRIC GROUPS CROSSED WITH DEVELOPMENTAL DATA GROUPS

The purpose of this section is to provide a more detailed view of the groups defined in the previous section by comparing groups formed by selected metrics with groups formed by the developmental data. The results presented in this section were obtained from the formation of two-way crosstabulation tables generated by the Statistical Analysis System (SAS). It is useful to present crosstabulation tables since they show combined frequency distributions for two variables. By observing the frequency distribution, the relationship between the two measures can be better identified. Furthermore, it is possible to isolate any anomalous components. For example, those components which belong to a high ERROR group, but also belong to a low INVOKE group would be considered anomalies. In addition to displaying the frequency distributions, SAS also outputs row, column and table percentages for each entry in a given crosstabulation table. These percentages will be used in this section to identify trends in the group level data.

For this analysis, it was necessary to also define group boundaries for ERRORS, WEIGHTED CHANGES, and CODING TIME. Since each of the three grouping techniques generally identified the same trend in the data, only one grouping method will be used in this analysis. The Logged Data technique,

suggested by Crawford et.al., was chosen as a representative grouping method since it is more sophisticated than a simplistic rank grouping (ie. uses the distances between components) and it attempts to compensate for the asymmetric properties inherit in the data.

The following three tables contain two-way crosstabulations between selected metric groups(LOC, INFO-LOC, INVOKE) and groups defined by the ERROR data. Tables for only these three metrics are presented since they more clearly indicate trends in the data.

Each table entry contains four numbers. From top to bottom, the first number represents the frequency count, the second number represents the table percentage, the third number represents the row percentage and the final number represents the column percentage. For example, in table 12 where LOC groups are crossed with ERROR groups, there were nine components which were classified as group 1 components for both the LOC metric groupings and the ERROR data groupings. These nine components represented 1.60% of the population, 90% of all components classified as ERROR group 1 components, and 3.85% of all components classified as LOC group 1 components.

The crosstabulation data given in table 12, table 13, and table 14 indicate some interesting relationships between the metric based groups and the ERROR based groups. First, notice that the row percentage for ERROR group 1 components

TABLE 12

LOC Groups Crosstabulated With ERROR Groups

LOC GROUPS / FREQ / PERCENT / ROW PCT / COL PCT	ERROR GROUPS 1	2	3	4	5	6	TOTAL
1	9	1	0	0	0	0	10
	1.60	0.18	0.00	0.00	0.00	0.00	1.78
	90.00	10.00	0.00	0.00	0.00	0.00	
	3.85	0.86	0.00	0.00	0.00	0.00	
2	42	3	3	6	1	0	55
	7.49	0.53	0.53	1.07	0.18	0.00	9.80
	76.36	5.45	5.45	10.91	1.82	0.00	
	17.95	2.59	3.95	7.89	2.13	0.00	
3	120	45	21	15	9	1	211
	21.39	8.02	3.74	2.67	1.60	0.18	37.61
	56.87	21.33	9.95	7.11	4.27	0.47	
	51.28	38.79	27.63	19.74	19.15	8.33	
4	45	55	34	32	23	7	196
	8.02	9.80	6.06	5.70	4.10	1.25	34.94
	22.96	28.06	17.35	16.33	11.73	3.57	
	19.23	47.41	44.74	42.11	48.94	58.33	
5	18	11	17	20	12	4	82
	3.21	1.96	3.03	3.57	2.14	0.71	14.62
	21.95	13.41	20.73	24.39	14.63	4.88	
	7.69	9.48	22.37	26.32	25.53	33.33	
6	0	1	1	3	2	0	7
	0.00	0.18	0.18	0.53	0.36	0.00	1.25
	0.00	14.29	14.29	42.86	28.57	0.00	
	0.00	0.86	1.32	3.95	4.26	0.00	
TOTAL	234	116	76	76	47	12	561
	41.71	20.68	13.55	13.55	8.38	2.14	100.00

typically decreases as the metric group number increases. The ERROR group 1 components are precisely those components

TABLE 13

INFO-LOC Groups Crosstabulated With ERROR Groups

INFO-LOC GROUP / FREQ / PERCENT / ROW PCT / COL PCT	ERROR GROUP 1	2	3	4	5	6	TOTAL
1	13	0	2	1	0	0	16
	2.32	0.00	0.36	0.18	0.00	0.00	2.85
	81.25	0.00	12.50	6.25	0.00	0.00	
	5.56	0.00	2.63	1.32	0.00	0.00	
2	44	14	4	7	8	0	77
	7.84	2.50	0.71	1.25	1.43	0.00	13.73
	57.14	18.18	5.19	9.09	10.39	0.00	
	18.80	12.07	5.26	9.21	17.02	0.00	
3	95	36	16	14	9	2	172
	16.93	6.42	2.85	2.50	1.60	0.36	30.66
	55.23	20.93	9.30	8.14	5.23	1.16	
	40.60	31.03	21.05	18.42	19.15	16.67	
4	70	49	35	26	16	3	199
	12.48	8.73	6.24	4.63	2.85	0.53	35.47
	35.18	24.62	17.59	13.07	8.04	1.51	
	29.91	42.24	46.05	34.21	34.04	25.00	
5	12	17	18	27	12	5	91
	2.14	3.03	3.21	4.81	2.14	0.89	16.22
	13.19	18.68	19.78	29.67	13.19	5.49	
	5.13	14.66	23.68	35.53	25.53	41.67	
6	0	0	1	1	2	2	6
	0.00	0.00	0.18	0.18	0.36	0.36	1.07
	0.00	0.00	16.67	16.67	33.33	33.33	
	0.00	0.00	1.32	1.32	4.26	16.67	
TOTAL	234	116	76	76	47	12	561
	41.71	20.68	13.55	13.55	8.38	2.14	100.00

which have been found to be error free. These percentages indicate that a larger fraction of components have errors

TABLE 14

INVOKE Groups Crosstabulated With ERROR Groups

INVOKE GROUPS / FREQ / PERCENT / ROW PCT / COL PCT	ERROR GROUPS 1	2	3	4	5	6	TOTAL
1	54	16	7	11	6	0	94
	9.63	2.85	1.25	1.96	1.07	0.00	16.76
	57.45	17.02	7.45	11.70	6.38	0.00	
	23.08	13.79	9.21	14.47	12.77	0.00	
2	40	12	5	3	1	0	61
	7.13	2.14	0.89	0.53	0.18	0.00	10.87
	65.57	19.67	8.20	4.92	1.64	0.00	
	17.09	10.34	6.58	3.95	2.13	0.00	
3	65	43	15	10	4	1	138
	11.59	7.66	2.67	1.78	0.71	0.18	24.60
	47.10	31.16	10.87	7.25	2.90	0.72	
	27.78	37.07	19.74	13.16	8.51	8.33	
4	65	38	37	38	21	1	200
	11.59	6.77	6.60	6.77	3.74	0.18	35.65
	32.50	19.00	18.50	19.00	10.50	0.50	
	27.78	32.76	48.68	50.00	44.68	8.33	
5	10	7	12	13	14	8	64
	1.78	1.25	2.14	2.32	2.50	1.43	11.41
	15.63	10.94	18.75	20.31	21.88	12.50	
	4.27	6.03	15.79	17.11	29.79	66.67	
6	0	0	0	1	1	2	4
	0.00	0.00	0.00	0.18	0.18	0.36	0.71
	0.00	0.00	0.00	25.00	25.00	50.00	
	0.00	0.00	0.00	1.32	2.13	16.67	
TOTAL	234	116	76	76	47	12	561
	41.71	20.68	13.55	13.55	8.38	2.14	100.00

for the higher metric groups than they do for the lower metric groups. For example, 90% of the components found in LOC

Group 1 were not reported to have any errors. However, only 76% of the components in LOC Group 2 were found to be errorless, 56% percent for LOC Group 3 and so on. Interestingly, all components belonging to the highest LOC group did contain at least one error. The results presented in table 15 show the percentage of components within each group which were found to contain no errors for all ten software metrics.

TABLE 15

Percentage of Errorless Procedures Across Groups

	GROUP NUMBER					
	1	2	3	4	5	6
LOC	90%	76%	56%	22%	21%	0%
CYCLO	76%	63%	53%	26%	25%	0%
EFFORT	80%	64%	54%	29%	18%	33%
INFOFLOW	73%	51%	53%	36%	17%	0%
STABILITY	60%	51%	40%	38%	31%	---
REVIEW	14%	---	53%	36%	17%	0%
INVOKE	57%	65%	47%	32%	15%	0%
STAB-LOC	56%	49%	40%	35%	26%	80%
REVIEW-LOC	20%	80%	61%	22%	22%	11%
INFO-LOC	81%	57%	55%	35%	13%	0%

This trend is also everywhere consistent for McCabe's CYCLO metric ranging from 76%(Group 1) down to 0%(Group 6). The third code metric, Halstead's EFFORT generally exhibits this same property, but does contain an anomalous jump between the Group 5 and Group 6 boundaries. Furthermore, of the four structure metrics, only Woodfield's Review complexity fails to show any pattern. The strongest relationship between the various structure metric groups and fraction of errorless components exist with the INFOFLOW groups. These percentages are steadily decreasing, ranging from 73%(Group 1) to 53%(Group 3) to 0%(Group 6). Of the three hybrid metrics, only the INFO-LOC metric defines groups with consistently decreasing percentages. The REVIEW-LOC metric exhibits this desirable trend for Group 2 through Group 6, however it does report only 20% of the components to be errorless for its least complex group of components.

A second relationship between the metric based groups and the ERROR based groups, can be inferred from the crosstabulation data. In the three crosstabulations presented, the upper right portion of each table is relatively sparse. This indicates that those components which were highly error prone, were typically the most complex components of the system. In particular, notice that eleven of the twelve components with the highest number of reported errors were located in the higher three groups for both LOC metric and the INVOKE metric. The information flow based metric, INFO-LOC

also performed quite well, indentifying ten of the twelve most error prone components. Table 16, found below summarizes the ability of each of the ten metrics to classify the error prone components as more complex. The entries found in this table represents the percentage of error prone components (Error groups 4,5 & 6) which were classified as the most complicated (Error groups 4,5 & 6). The percentages in table 16 indicate that the code metrics identify a higher percentage of the error prone components than either of the other two classes of metrics. Of the remaining seven metrics, McClure's INVOKE metric and both the information based metrics identify approximately 70% of the most error prone components as the more complicated.

A third trend in the crosstabulation data is graphically displayed in the form of "city block" charts. A few of these charts are illustrated in figure 1 and figure 2 found below. Each of the following city block charts contain "buildings" representing the row percentages found in associated two-way crosstabulation tables. Observe the behavior of percentage distributions across metric groups. For example, consider the block chart found in figure 1 (a). Notice that 90% of all LOC Group 1 components also reside in ERROR Group 1. However 10% of the LOC Group 1 components have shifted into ERROR group 2. Notice further that only 76.36% of all LOC Group 2 components reside in ERROR Group 1. The remaining 23.64% of the components have "seeped" into

TABLE 16

Percentage of Error Prone Components Identified as Complex

Metric	Percentage
LOC	76%
CYCLO	74%
EFFORT	77%
INVOKE	73%
INFOFLOW	69%
STABILITY	61%
REVIEW	47%
INFO-LOC	70%
STAB-LOC	63%
REVIEW-LOC	59%

ERROR groups 2 through 5. More and more components seep out of ERROR Group 1 as the LOC group number increases. This trend was previously highlighted in table 12. However, this city block chart graphically emphasizes that as the LOC group number increases, more and more components generally seep into the higher ERROR groups. Ultimately, for LOC Group 6, all the components have seeped into the higher ERROR Groups with 71.33% of the components residing in either ERROR Group 4 or ERROR Group 5. It is somewhat surpris-

ing that none of the LOC Group 6 components seeped into the highest ERROR group. Notice that this was not the case for the other three city block charts found in figure 1. These three charts indicate that McClure's INVOKE measure and both of Henry's Information Flow measures also exhibit the seeping effect. The rate in which components seep into the higher ERROR groups appears to be somewhat gradual for the first three or four metric based groups. This is illustrated by the line which is superimposed on each of the four city block charts. This line, connecting the tallest buildings within each of the metric based groups, roughly indicates the rate of seepage across the six metric based groups. Notice that the line does not jump sharply until the transition from Group 4 to Group 5 is made. The gradual seepage of components for the first four groups, followed by larger amounts of seepage for the last two groups is consistent with the previous analysis using the LSD multiple comparison test. Recall that Fisher's Protected LSD test indicated that the ERROR means of the lower complexity groups were not found to be significantly different but did declare significant differences to exist for the ERROR means of the higher complexity groups.

Figure 1: Row Percentages Between Metrics and Error Groups

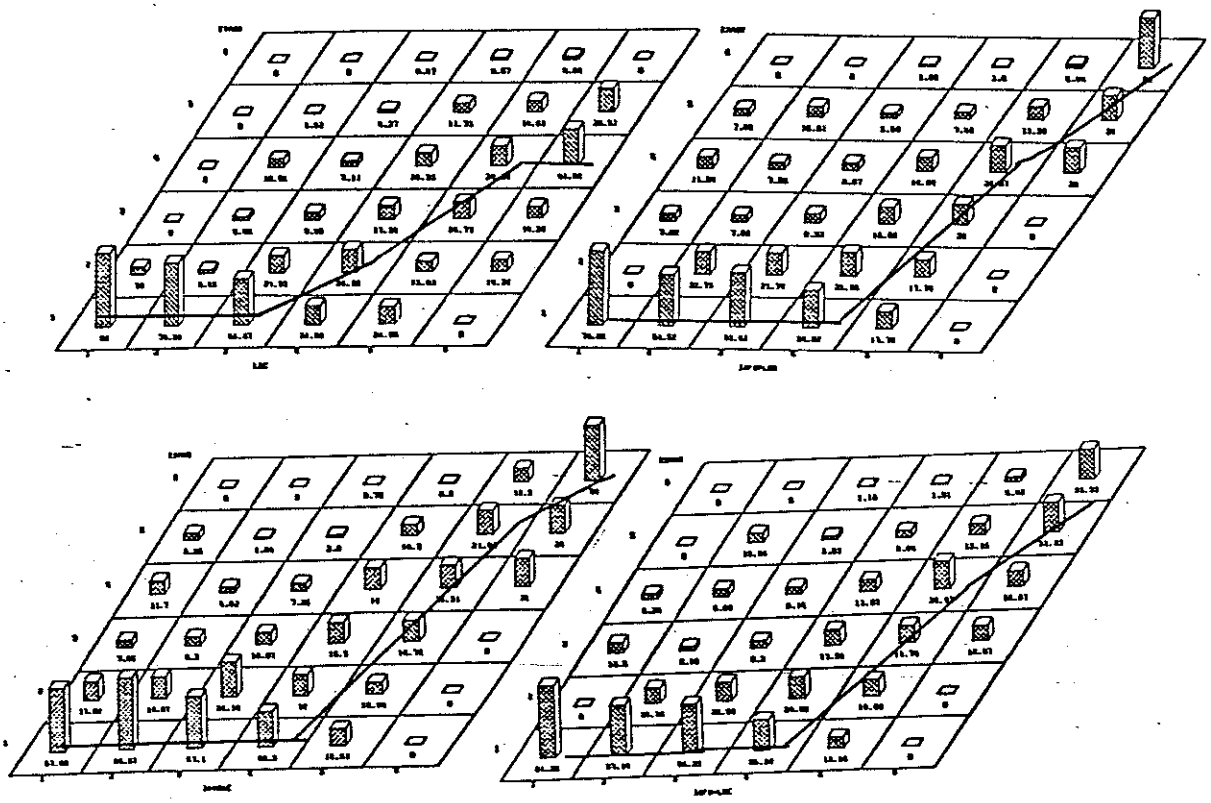
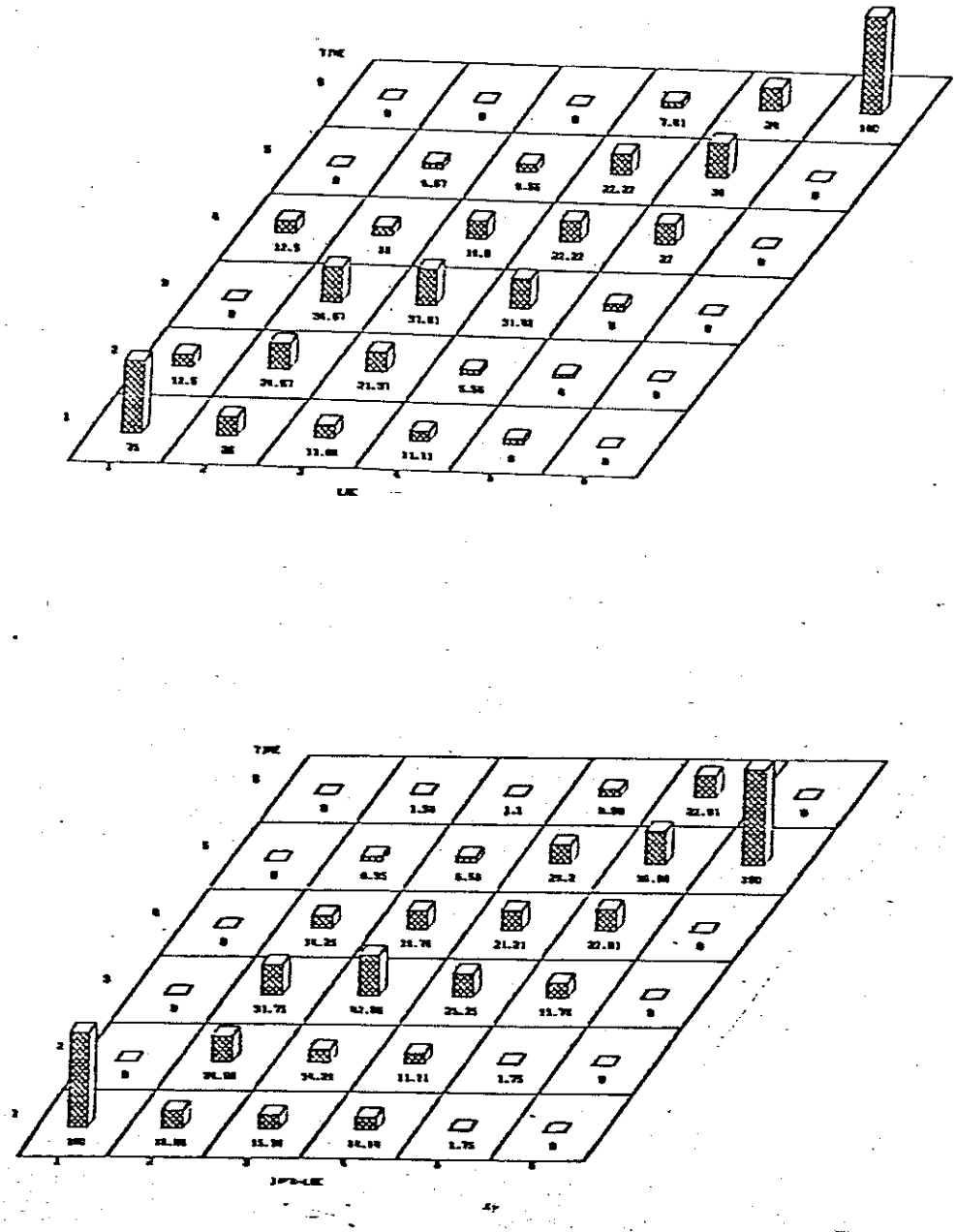


Figure 2: Row Percentages Between Metrics and Time Groups



The two city block charts comprising figure 2 graph the row percentages found in the crosstabulations between groups based on CODING TIME and groups based on both the LOC metric and the INFO-LOC metric. Both of the block charts found in figure 2 also exhibit a shift in the distribution of percentages across the metric based groups. Each of these diagrams illustrate the fact that components belonging to higher complexity based groups will generally take longer to implement. The trend is particularly noticeably in figure 2 (a), where the tallest buildings all lie on the main diagonal.

Finally, the information contained in the two-way crosstabulations between the metric based groups and the WEIGHTED CHANGES based groups has been summarized in the following table. The data in table 17 captures the ability of the metrics to identify components which will be highly affected by changes in the system. To obtain the percentages given in this table, the components were partitioned into two classes: severely impacted components and weakly impacted components. Severely impacted components are precisely those routines which possessed the highest WEIGHTED CHANGE values. In particular, a component possessed a "high" WEIGHTED CHANGE value if the Logged Data grouping method positioned the component into either Group 4, Group 5, or Group 6. The entries in table 10 indicate the percentage of components within each metric based group, which

were severely impacted by changes in the system. For example,72% of the components in LOC group 5 were also identified to be severely impacted components, while 100% of the components in INVOKE group 6 were severely impacted components. The percentages given in the last two columns of table 10 generally indicate that components placed into the higher metric groups are likely to be severely impacted by system changes. Only the group 6 percentage for the STAB-LOC metric falls below 56%. Furthermore, the three code metrics indicate that components placed in the lower metric groups are not as likely to be severly impacted. Unfortunately, the structure metrics do not show a similar trend. Notice that for group 1 components, each of the structure metrics still report rather high percentages of severely impacted components, ranging from 36 to 70 percent.

TABLE 17

Fraction of Severely Impacted Components

	GROUP NUMBER					
	1	2	3	4	5	6
LOC	0%	35%	43%	54%	72%	100%
CYCLO	15%	25%	43%	63%	66%	100%
EFFORT	5%	39%	48%	50%	81%	62%
INFOFLOW	58%	33%	41%	60%	77%	100%
STABILITY	40%	35%	47%	55%	63%	---
REVIEW	70%	---	44%	53%	56%	0%
INVOKE	36%	31%	40%	60%	82%	100%
STAB-LOC	29%	50%	49%	58%	65%	20%
REVIEW-LOC	0%	29%	44%	51%	70%	100%
INFOFLOW LOC	55%	34%	42%	45%	85%	100%

PART VI

SUMMARY AND CONCLUSIONS

This paper has reported on experiments which identified and compared groups of components. Three different grouping techniques: Logged Data, Clustering and Rank, were employed, each of which generally produced similar results. It was found that many of the metrics were able to partition the components into groups which possessed significantly different means for ERRORS, WEIGHTED CHANGES, and CODING TIME. Furthermore, the two information flow metrics and McClure's invocation complexity were particularly able to separate out those components which possessed extremely high developmental data readings.

This paper also presented results from two-way crosstabulations between the component groups formed by the metrics and component groups formed by the observed measures of complexity. One result of these crosstabulations indicated the relationship between metric based groups and their percentage of errorless components. It was found that some metrics: LOC, CYCLO, INFOFLOW, STABILITY and INFO-LOC were able to define increasingly more complex groups which had a decreasing percentage of errorless components. Furthermore, the six metrics: LOC, CYCLO, EFFORT, INFOFLOW, INVOKE, AND INFO-LOC were able to identify approximately seventy percent of the most error prone components.

The crosstabulation data was also graphically displayed in the form of "city block" charts which revealed percentage distributions of the data across selected metric groups. These distributions once again illustrated that as more complex groups are considered, a larger fraction of the population will shift into the higher ERROR or CODING TIME groups. top on

REFERENCES

1. [Basi83]Basili,V.,Selby R.,Phillips T., "Metric Analysis and Data Validation Across Fortran Projects" IEEE Transactions on Software Engineering, Vol. SE-9,No. 6. November 1983

2. [Cann85]Canning J. Applying Structure and Code Metrics to Large Scale Systems Ph.d dissertation Virginia Polytechnic and State University, June 1985

3. [Curt79]Curtis,W. , Sheppard,S.B. , and Milliman,P.M. "Third Time Charm : Stronger Prediction of Programmer Performance by Software Complexity Metrics , " Fourth International Conference on Software Engineering September 1979 , Munich , Germany , 356-360 .

4. [Hals77]Halstead,M.H. Elements of Software Science , Elsevier North-Holland, Inc. , New York , NY , 1977.

5. [Hame81]Hamer, Peter, Frewin, Gillian, "M.H. Halstead's Software Science - A Critical Examination," ITT Technical Report No. STL 1341, July 1981.

6. [Hane72]Haney, Frederick, "Module Connection Analysis - A Tool for Scheduling Software Debugging Activities,"

7. [Henr79]Henry,Sallie, Information Flow Metrics for the Evaluation of Operating Systems' structure Ph.d Dissertation,Iowa State University, 1979.

8. [Henr81a]Henry,Sallie and D. Kafura, "Software Structure Metrics Based on Information Flow", IEEE Transactions on Software Engineering, Vol. SE-7, No. 5, pp. 510-518, September, 1981.

9. [Henr81b]Henry, Sallie, D. Kafura, and K. Harris, "On the Relationships Among Three Software Metrics", Performance Evaluation Review, Vol. 10, No. 1, pp. 81-88 Spring 1981.

10. [Henr84]Henry, Sallie and D. Kafura, "The Evaluation of Software Systems' Structure Using Quantitative Software Metrics", Software: Practice and Experience Vol. 14(6) June 1984 pp.561-573.

11. [Kafu82]Kafura,D., Henry,S., "Software Quality Metrics Based on Interconnectivity," The Journal of Systems and Software , Vol. 2 , pp. 121-131, 1981.

12. [Kafu84]Kafura D., Canning J., and Reddy G., "The Independence of Software Metrics Taken at Different Life-Cyle Stages" Proceedings: Ninth Annual Software Engineering Works Goddard Space Flight Center, Nov. 28, 1984

13. [Kafu85a]Kafura D.,Canning J., "A Validation of Software Metrics Using Many Metrics and Many Resources" Proceedings of the International Conference of Software Engineering August 1985

14. [Kafu85b]Kafura D.,Canning J., "Applying Structure and Code Metrics to Three Large Scale Systems." Journal of Systems and Software, submitted August 1985.

15. [Lync81]Lynch, W.C., and J.C. Browne, "Performance Evaluation: A Software Metrics Success Story", Journal of Systems and Software, Vol. 2, pp. 105-112, 1981. [McCa76]McCabe,T,J, "A Complexity Measure , " IEEE Transactions on Software Engineering , SE-2 , 4 (December 1976) , 308-320.

16. [McCl78]McClure, C. "A Model for Program Complexity Analysis," Proceedings Third International Conference on Software Engineering , Atlanta, Ga. May 1978 , 149-157.

17. [Musa75]Musa,J.D. "A Theory of Software Reliability and Its Application," IEEE Transactions on Software Engineering, vol.SE-1, no. 3, pp. 312-327, 1975

18. [Nasa81] National Aeronautics and Space Administration, "Software Engineering Laboratory (SEL) Data Base Organization and User's Guide", Software Engineering Laboratory Series SEL-81-002, Sept. 1981

19. [Ross77]Ross,Douglas T., "Structured Analysis (SA): A Language for Communicating Ideas," IEEE Transactions on Software Engineering, Jan. 1977.

20. [Wood80]Woodfield,S.N., Enhanced Effort Estimation by Extending Basic Programming Models to Include Modularity Factors, Ph. D. Thesis, Purdue University, Computer Science Dept., 1980.

21. [Yau80]Yau S., Collofello J., "Some Stability Measures for Software Maintenance," IEEE Transactions on Software Engineering, Vol. SE-6,No.6,Nov.1980.