



Calhoun: The NPS Institutional Archive

Reports and Technical Reports

All Technical Reports Collection

1997

Integration of Software Metrics with Quality and Reliability

Schneidewind, Norman F.

<http://hdl.handle.net/10945/25624>



Calhoun is a project of the Dudley Knox Library at NPS, furthering the precepts and goals of open government and government transparency. All information contained herein has been approved for release by the NPS Public Affairs Officer.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

<http://www.nps.edu/library>

NAVAL POSTGRADUATE SCHOOL Monterey, California



MCTSSA Software Reliability Handbook

Volume III

Integration of Software Metrics with Quality and Reliability

by

Norman F. Schneidewind

18 June 1997

Approved for public release; distribution is unlimited.

Prepared for: U.S. Marine Corps
Tactical Systems Support Activity
Camp Pendleton, CA 92244-5171

13370318 133

NAVAL POSTGRADUATE SCHOOL
Monterey, California

RADM M.J. Evans
Superintendent

Richard Elster
Provost

This report was prepared for and funded by the U.S. Marine Corps, Systems Support Activity, Camp Pendleton, CA 92255-5171.

Reproduction of all or part of this report is authorized.

This report was prepared by:



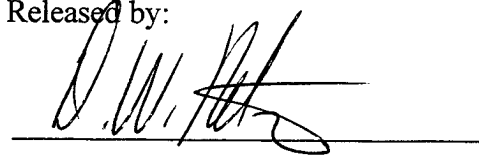
Norman F. Schneidewind
Department of Systems Management

Reviewed by:



Reuben T. Harris, Chairman
Systems Management Department

Released by:



D.W. Netzer, Associate Provost and
Dean of Research

REPORT DOCUMENTATION PAGE

Form Approved

OMB No 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)

2. REPORT DATE

18 June 1997

3. REPORT TYPE AND DATES COVERED

Technical Report

4. TITLE AND SUBTITLE

MCTSSA Software Reliability Handbook

Volume III

Integration of Software Metrics with Safety and Reliability

5. FUNDING

RLACH

6. AUTHOR(S)

Dr. Norman F. Schneidewind

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Department of Systems Management

Naval Postgraduate School

Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION

REPORT NUMBER

NPS-SM-97-004

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

U.S. Marine Corps Tactical Systems Support Activity

Box 555171 Building 31345

Camp Pendleton, CA 92255-5171

10. SPONSORING/MONITORING

AGENCY REPORT NUMBER

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

14. SUBJECT TERMS

15. NUMBER OF PAGES

39

16. PRICE CODE

17. SECURITY CLASSIFICATION
OF REPORT
UNCLASSIFIED18. SECURITY CLASSIFICATION
OF THIS PAGE
UNCLASSIFIED19. SECURITY CLASSIFICATION
OF ABSTRACT
UNCLASSIFIED20. LIMITATION OF
ABSTRACT
SAR

MCTSSA SOFTWARE RELIABILITY HANDBOOK

VOLUME III

INTEGRATION OF SOFTWARE METRICS WITH QUALITY AND RELIABILITY

18 June 1997

Revised: 15 July 1997

Dr. Norman F. Schneidewind

Code SM/Ss
Naval Postgraduate School
Monterey, CA 93943

Voice: (408) 656-2719

Fax : (408) 656-3407

Email: schneidewind@nps.navy.mil

DMIC QUALITY INSPECTED 8

TABLE OF CONTENTS

INTRODUCTION	4
Definitions	5
OBJECTIVES	5
DISCRIMINATIVE POWER MODEL	
.....	6
Discriminative Power Validation	
.....	6
Quality Control	6
Discriminative Power	6
DISCRIMINATIVE POWER VALIDATION MODEL	
.....	7
Stage 1: Identify Candidate Metrics	
.....	7
Stage 2: Compute Critical Values	
.....	8
Stage 3: Perform Contingency Table Analysis	
.....	9
Application Contingency Table	
.....	10
Statistical Validation	
.....	11
Misclassification	11
Application Criteria	
.....	13
Quality	13
Inspection	14
Summary of Validation Results	
.....	14
Stage 4: Apply a Stopping Rule for Adding Metrics	
.....	15
COMPARISON OF VALIDATION WITH APPLICATION RESULTS	
.....	15
PREDICTABILITY VALIDATION MODEL	
.....	16
Module Counts	17
Quality Factor Counts	
.....	18

INTEGRATION OF METRICS WITH QUALITY AND RELIABILITY	20
CONCLUSIONS	22
References	22
APPENDIX A. SAMPLE STATGRAPHICS METRICS ANALYSIS SESSION	25
DISCRIMINATIVE POWER VALIDATION MODEL	25
PREDICTABILITY VALIDATION MODEL	36

INTRODUCTION

Metrics should be collected for a purpose. A major purpose is to provide early indicators of software quality problems which may occur later in the project. You want to correct these problems early in the project when the cost is low. Thus there should be an integration of metrics, as early indicators, and the software reliability that is achieved during test and operation. To support this objective, this model has been developed to validate and apply metrics for quality control and quality prediction, with the primary objective of using metrics as early indicators of software quality and reliability problems. Metrics and quality factor data from the *Space Shuttle* flight software are used as an example. The approach is to integrate quality control and prediction in a single model and to validate metrics with respect to a quality factor in accordance with the metrics validation methodology developed in [SCH92] and included in the *IEEE Standard for a Software Quality Metrics Methodology (1061)* [IEE93]. Although a quality factor is a direct measurement of software quality and, hence, of more interest to customers and users than metrics, quality factors cannot be collected early in a project. Thus the need arises to validate metrics, which developers can collect early in a project, to act as an indirect measurement for the quality factor.

Boolean discriminant functions (BDFs) have been developed for use in the quality control and prediction process. These functions provide good accuracy (i.e., $\leq 3\%$) for classifying low quality software. This is true because the BDFs consist of more than just a set of metrics. They include additional information for discriminating quality: critical values. Critical values are threshold values of metrics that are used to either accept or reject modules when the modules are inspected during the quality control process. Note that to reject a module does not mean to discard it. Rather it means that the module receives priority attention to see whether its metric's values are a natural consequence of the functionality of the module. If this is not the case, then the module may be a candidate for redesign or the design may be left intact and the module given priority treatment during inspection and testing. Note that critical values should be determined quantitatively and that they can be highly application and project dependent.

A series of nonparametric statistical methods is used to: 1) identify a set of candidate metrics for further analysis, 2) identify the critical values of the metrics, and 3) find the optimal function of metrics and critical values based on the ability of the BDF to satisfy *both* statistical and application (i.e., quality and cost) criteria. The key to the approach is the use of validated metrics for early identification and resolution of quality problems.

In order to investigate the feasibility of validating and applying metrics for controlling quality on large software projects, metrics are validated on one random sample of modules and applied to three random samples that are both disjoint among themselves and from the validation sample, drawn from a population of 1397 modules of *Space Shuttle* flight software.

It is important to perform a marginal analysis when making a decision about how many metrics should be used in quality control and prediction processes. If many metrics are added at once, the contribution of individual metrics is obscured. Also, the marginal analysis provides an effective stopping rule for deciding when to stop adding metrics. Certain metrics are dominant in their effects on classifying quality (i.e., dominant metrics make fewer mistakes in classifying metrics than non-dominant ones) and that additional metrics are not needed to accurately classify quality. This effect is called *dominance*. Related to the property of *dominance* is the property of *concordance*, which is the degree to which a set of metrics produces the same result in classifying software quality. A high value of *concordance* implies that additional metrics will

not make a significant contribution to accurately classifying quality; hence, these metrics are redundant. For example, contrary to what you would expect, the metrics *comments* and *statements*, when combined in a single BDF, were better indicators of the quality of the *Space Shuttle* software than complexity metrics.

Now definitions are provided that are necessary for understanding the metrics model in general and the *Space Shuttle* flight software measurement environment in particular. This is followed by a description of objectives. Then the *Discriminative Power* model and its approach to validation are explained. Next validation results are compared with application results for quality control. Then the quality prediction part of the integrated model is developed and validation and application results are compared. Last metrics are integrated with quality and reliability. Appendix A contains a sample *Statgraphics* metrics analysis session.

Definitions

Quality Factor: An attribute of software that contributes to its quality [SCH92] where quality is the degree to which software meets customer or user needs or expectations [IEE90]. For example, reliability, an attribute that contributes to quality, is a factor. Quality factors are *customer or user oriented*. They are attributes that customers or users expect to see in the delivered software. Ideally, you want to directly measure quality factors like *reliability*. Unfortunately, direct measurement of *reliability*, for example, by *absence of failure during the specified mission*, can only be obtained late in a software project during the test and operations phases. Therefore the project may have to resort to an indirect measurement of quality, such as number of discrepancy reports (DRs) written against modules, *drcount*, where the DRs record deviations from requirements, as is the case with the *Space Shuttle* flight software. Thus, in this example, *drcount* will be treated as a quality factor.

Quality Metric: A function (e.g., cyclomatic complexity $M=e-n+2p$) whose inputs are software data (elementary software measurements, such as number of edges e , number of nodes n , and number of connected components p , in a directed graph) and whose output is a single numerical value M that can be interpreted as the degree to which software possesses a given attribute (cyclomatic complexity) that may affect its quality (e.g., reliability) [IEE93, SCH92]. A special case is the identity function wherein the software data (e.g., nodes) are used as metrics. This is the case in the *Space Shuttle* flight software application.

OBJECTIVES

One objective of this handbook is to show that there are relationships between a quality factor and metrics that allow you to control and predict the quality of software on large-scale projects such as the *Space Shuttle flight* software. In this application the quality factor is number of discrepancy reports *drcount* written against a module. A second objective is to integrate quality control and prediction in one model by deriving prediction equations from the *Contingency Table*, which provides the framework for the control function. Quality control is achieved by using metrics in BDFs during design to see whether the quality of the product is within the thresholds of acceptable quality, and to indicate whether remedial action is necessary (e.g., detailed inspection and tracking of the quality of the product during test and operation). Quality prediction involves computing point estimates and confidence limits of various quality and cost quantities to give software managers forecasts of quality and the cost to achieve it.

The Boolean discriminant function (BDF) is a new type of discriminant for classifying software quality

and the application of Kolmogorov-Smirnov (K-S) distance is a new way to determine a metric's critical value developed by the author. The author has also developed a new stopping rule for adding metrics: the ratio of the relative improvement in quality to the relative increase in cost.

DISCRIMINATIVE POWER MODEL

Discriminative Power Validation

Using the metrics validation methodology [SCH92], and the *Space Shuttle* flight software metrics and discrepancy reports, it is shown how to validate metrics with respect to the quality factor *drcount*. In brief, this involves conducting statistical tests to determine whether there is a high degree of association between *drcount* and candidate metrics.

Quality Control

The quality control function is applied to the Application Sample to flag software for detailed inspection that is below quality limits. Quality control is the evaluation of modules with respect to predetermined critical values of metrics. The purpose of quality control is to allow software managers to identify software that does not meet quality requirements early in the development process so corrective action can be taken when the cost is low. The *Discriminative Power* validity criterion is applied to the Validation Sample to validate metrics which will subsequently be used to control the quality of the Application Sample. *Discriminative Power* is defined as follows [SCH92]:

Discriminative Power

Given matrix M_{ij} of n modules and m metrics (i.e., nm metric values), vector M_{cj} of m metric critical values, vector F_i of n quality factor values, and scalar F_c of quality factor critical value, M_{ij} must be able to discriminate with respect to F_i , for a specified F_c , as shown in the following relation:

$$\begin{aligned} M_{ij} > M_{cj} &\Leftrightarrow F_i > F_c \text{ and} \\ M_{ij} \leq M_{cj} &\Leftrightarrow F_i \leq F_c \end{aligned} \tag{1}$$

for $I=1,2,\dots,n$, and $j=1,2,\dots,m$ with specified α , where α is the significance level of a statistical test for estimating the degree to which (1) holds. In other words do the indicated metric relations imply corresponding quality factor relations in (1)?

This criterion assesses whether M_{cj} **has** sufficient *Discriminative Power* to be capable of distinguishing a set of high quality modules from a set of low quality modules. If this is the case, use the critical values to flag modules of the Application Sample that appear to have unacceptable or questionable quality.

The desired quality level is set by the choice of F_c , by validating M_{cj} with respect to F_c . If a low value of F_c (e.g., low *drcount* implying high quality) is selected, it would produce an M_{cj} that would flag more modules than would be the case if a high value of F_c (high *drcount* implying low quality) were selected.

Having defined the *Discriminative Power* validity criterion, a model is developed that will allow you to *validate* metrics for controlling quality during software design. In order to validate metrics, it is necessary

to collect both quality factor data and metric data. This will only be feasible when development has progressed to the point where quality factor data, such as discrepancy reports, are available (i.e., inspections have been conducted or tests have been held). In contrast, when metrics are *applied* during design, only the metric data are available. Thus metrics are used as estimates of quality until the point in development is reached when the quality factor data are available. Also it is important to recognize that validation is performed retrospectively. That is, with both metrics and quality factors in hand, you can evaluate how well the metrics would have performed if they had been applied to the Validation Sample. If the metrics perform well, then they are validated and you would expect that they will perform adequately when applied to the Application Sample (not as well as when applied to the Validation Sample because of possible differences in module characteristics between the Validation and Application samples) but better than if unvalidated metrics are used.

DISCRIMINATE FILTER VALIDATION MODEL

The basis of this model is a methodology for validating BDFs and their critical values that have the ability to discriminate high quality from low quality. Now use a three stage process for selecting metrics for quality control: 1) identify candidate metrics; 2) compute critical values of the candidate metrics; and 3) for the set of candidate metrics and critical values, find the optimal combination based on statistical and application (i.e., quality and cost) criteria.

1) Stage 1: Identify Candidate Metrics

In Stage 1 rank the metrics according to their ability to discriminate between two samples of modules: $F_i \leq F_c$ versus $F_i > F_c$, which correspond to $drcount=0$ and $drcount>0$, respectively, in the *Space Shuttle* flight software. This analysis is independent of the metrics' critical values (i.e., only the ranks of the metrics are used). The Kruskal-Wallis One-Way Analysis By Ranks (K-W) is used for this analysis. Data from the two samples are ordered and ranked and mean ranks are computed for the sets. If the difference in mean ranks is significant, as determined by the value of the test statistic and significance level (i.e., $\alpha \ll .005$), you would conclude that the corresponding populations differ; thus, metrics that pass this test are classified as *candidate metrics*.

The thirteen metrics that were collected with the quality factor *drcount*, for 1397 modules written in HAL/S, are defined in Table 1. The results of the K-W test for the Validation Sample are shown in Table 2; a bar diagram is shown in Figure 1. Use the K-W test to identify the set of candidate metrics because: 1) K-W has proven to be a good indicator of the relative ability of metrics to classify quality; 2) experience has shown that only a subset of the collected metrics is necessary to identify the optimal BDF, as described in Stage 3; and 3) the amount of computation involved for computing the Kolmogorov-Smirnov (K-S) distance for all metrics (see Stage 2) is large. Thus you use the procedure of screening the metrics initially, using K-W as the criterion, and including metrics for further analysis in Stage 2 and Stage 3 by selecting them from the top in Table 2 (i.e., *comments*), and continuing the process until the stopping rule in Stage 3 (to be described later) is satisfied.

		Symbol	
Metric	Definition (counts per module)	Metric	Critical Value
eta1	unique operator count	E1	E1 _c
eta2	unique operand count	E2	E2 _c
η1	total operator count	η1	η1 _c
η2	total operand count	η2	η2 _c
statements	total statement count (executable code; no comments)	S	S _c
loc	total non-commented lines of code	L	L _c
comments	total comment count	C	C _c
nodes	total node count (in control graph)	N	N _c
edges	total edge count (in control graph)	E	E _c
paths	total path count (in control graph)	P	P _c
maxpath	maximum path length (edges in control graph)	MP	MP _c
avepath	average path length (edges in control graph)	AP	AP _c
cycles	total cycle count (in control graph)	CY	CY _c

Stage 2: Compute Critical Values

Once the metrics have been ranked, critical values M_{c_j} are computed, using a new method which the author has developed, which is based on the Kolmogorov-Smirnov (K-S) test [CON71]. This method tests whether the sample cumulative distribution functions (CDF) are from the same or different populations. The test statistic is the maximum vertical difference between the CDFs of two samples (e.g., the CDFs of M_{ij} for $drcount \leq F_c$ and $drcount > F_c$). If the difference is significant (i.e., $\alpha \leq .005$), the value of M_{ij} corresponding to maximum CDF difference is used for M_{c_j} . This relationship is expressed in equation (2). This concept is illustrated in Figure 2, for the critical value of comments, where the CDFs for $drcount=0$ and $drcount>0$ are shown. In this example, $C_c=38$. This is the value of comments where there is the maximum difference between the CDFs. Figure 3 shows the *difference* in CDFs for comments, where the curve reaches a maximum at 38. Table 3 shows the metric critical values M_{c_j} and the K-S distances for the thirteen metrics of the Validation Sample. In addition, the ranks of K-W and K-S show that several metrics rank high with respect to both statistics.

$$K-S(M_{c_j}) = \max \{ [CDF(M_{ij}/(F_i \leq F_c))] - [CDF(M_{ij}/(F_i > F_c))] \} \quad (2)$$

Metrics are added to the BDF in the order of their K-S Distance.

Metric	Kruskal-Wallis			Kolmogorov-Smirnov			
	Statistic	α	Rank	Critical Value	Distance	α	Rank
comments	37.91	$\approx 10^{-10}$	1	38	0.585	0.005	1
eta1	26.03	$\approx 10^{-7}$	2	10	0.492	0.005	3
statements	25.60	$\approx 10^{-7}$	3	26	0.557	0.005	2
nodes	21.34	$\approx 10^{-5}$	4	11	0.487	0.005	4
edges	21.33	$\approx 10^{-5}$	5	10	0.487	0.005	4
loc	21.29	$\approx 10^{-6}$	6	19	0.415	0.005	9
maxpath	19.18	$\approx 10^{-5}$	7	8	0.452	0.005	6
paths	19.14	$\approx 10^{-4}$	8	1	0.475	0.005	5
avepath	18.83	$\approx 10^{-5}$	9	4	0.440	0.005	8
eta2	18.20	$\approx 10^{-5}$	10	33	0.451	0.005	7
η_1	16.57	$\approx 10^{-5}$	11	26	0.386	0.005	10
η_2	15.70	$\approx 10^{-5}$	12	21	0.386	0.005	10
cycles	6.59	$\approx 10^{-2}$	13	0	0.220	0.050	11

Stage 3: Perform Contingency Table Analysis

condition is expressed by a Boolean *OR* function of the metrics. This is the *REJECT* column, meaning that according to the classification decision made by the metrics, these modules have unacceptable quality. The top row contains modules that are high quality; these modules have a quality factor that does not exceed its critical value (e.g., zero *drcount*). The bottom row contains modules that are low quality; these modules have a quality factor that exceeds its critical value (e.g., *drcount*>0).

Equation (3) gives the module count, based on BDFs of F_i and M_j , that are calculated over the n modules for m metrics. This equation is an implementation of the relation given in (1).

$$\begin{aligned}
 C_{11} &= \text{COUNT FOR } ((F_i \leq F_c) \wedge (M_{i1} \leq M_{c1}) \dots \wedge (M_{ij} \leq M_{cj}) \dots \wedge (M_{im} \leq M_{cm})) \\
 &\quad \text{FOR } i=1 \text{ TO } n \\
 C_{12} &= \text{COUNT FOR } ((F_i \leq F_c) \wedge ((M_{i1} > M_{c1}) \dots \vee (M_{ij} > M_{cj}) \dots \vee (M_{im} > M_{cm}))) \\
 &\quad \text{FOR } i=1 \text{ TO } n \\
 C_{21} &= \text{COUNT FOR } ((F_i > F_c) \wedge (M_{i1} \leq M_{c1}) \dots \wedge (M_{ij} \leq M_{cj}) \dots \wedge (M_{im} \leq M_{cm})) \\
 &\quad \text{FOR } i=1 \text{ TO } n \\
 C_{22} &= \text{COUNT FOR } ((F_i > F_c) \wedge ((M_{i1} > M_{c1}) \dots \vee (M_{ij} > M_{cj}) \dots \vee (M_{im} > M_{cm}))) \\
 &\quad \text{FOR } i=1 \text{ TO } n
 \end{aligned} \tag{3}$$

for $j=1, \dots, m$, and where $\text{COUNT}(I) = \text{COUNT}(I-1) + 1$ FOR Boolean expression *true* and $\text{COUNT}(I) = \text{COUNT}(I-1)$, otherwise; $\text{COUNT}(0) = 0$.

The counts correspond to the cells of the *Contingency Table*, as shown in Table 3, where row and column totals are also shown: n , n_j , n_i , N_j , and N_i . The analysis could be generalized to include multiple quality factors, if necessary; in this case the *Contingency Table* would have more than two rows. Note that Table 3 is the *Contingency Table* for *validation* that contains both quality factor and metric data.

In addition to counting modules in Table 3, you must also count the quality factor (e.g., *drcount*) that is incorrectly classified. This is shown as Remaining Factor, RF, in the *ACCEPT* column. This is the quality factor count (e.g., *drcount*) on modules that should have been rejected. Also shown is Total Factor, TF, the total quality factor count on all the modules in the sample. Lastly Table 3 shows RFM (Remaining Factor Modules) that is the count of modules with quality factor count > 0 (i.e., modules with Remaining Factor, RF).

Table 3 and subsequent equations show an example validation, where the optimal combination of metrics from Table 1 and their critical values for a random sample of 100 modules (sample 1), from the population of 1397, is *comments* with a critical value of **38** and *statements* with a critical value of **26**. The critical values were obtained by using the ranking of metrics from the K-W test in Table 2 and applying the K-S criterion as given by equation (2) and illustrated in Figures 2 and 3. Later it will be explained how you would arrive at the particular combination of metrics as the optimal set. The reason that *comments* is the leading metric in this example is that in the case of the *Space Shuttle* the comments are not limited to the conventional comments of describing the purpose and action of the statements in a program. In addition to this type of comment, and more important, the *Space Shuttle comments* contain a history of the experience with a module: requirements, design approach, inspections, tests, discrepancy reports, etc. Thus, in general, more *comments*, means more problems with the software!

Application Contingency Table

A different Contingency Table is used for the application of validated metrics. This table contains only metrics data. This is shown in Table 4, where the "?" indicates that the quality factor data F_i is not available

when the validated metrics are used in the quality control function of the application project. During the design phase of the application project, modules are classified according to the criteria that have been described. A second disjoint random sample of 100 modules (sample 2) was used to illustrate the process. Whereas 31 and 69 modules were accepted and rejected, respectively during validation, 40 and 60 modules were accepted and rejected, respectively, during the application. The rejected modules would be given priority attention, as was described previously.

Statistical Validation

Validate M_{c_j} statistically by demonstrating that it partitions Table 3 in such a way that C_{11} and C_{22} are large relative to C_{12} , and C_{21} . If this is the case, a large number of high quality modules (e.g., modules with zero *drcount*) would have $M_{ij} \leq M_{c_j}$ and would be correctly classified as high quality. Similarly, a large number of low quality modules (e.g., modules with *drcount* > 0) would have $M_{ij} > M_{c_j}$ and would be correctly classified as low quality. The degree to which this is the case is estimated by the chi-square (χ^2) statistic. If calculated $\chi^2_c > \chi^2_s$ (chi-square at specified α_c) and if calculated $\alpha_c < \alpha_s$, conclude that M_{c_j} is statistically significant.

Misclassification

As part of the statistical validation, estimate the *Discriminative Power* of M_{c_j} by noting in Table 3 that ideally $C_{11} = n_1 = N_1$, $C_{12} = 0$, $C_{21} = 0$, $C_{22} = n_2 = N_2$. The extent that this is not the case is estimated by *Type 1* misclassifications (i.e., the module has **Low Quality** and the metrics "say" it has **High Quality**) and *Type 2* misclassifications (i.e., the module has **High Quality** and the metrics "say" it has **Low Quality**). Thus the following measures of misclassification are defined:

$$\text{Proportion of Type 1: } P_1 = C_{21}/n \tag{4}$$

For the example, $P_1 = (1/100) * 100 = 1\%$.

$$\text{Proportion of Type 2: } P_2 = C_{12}/n \tag{5}$$

$P_2 = (27/100) * 100 = 27\%$.

$$\text{Proportion of Type 1+Type 2: } P_{12} = (C_{21} + C_{12})/n \tag{6}$$

$P_{12} = ((1+27)/100) * 100 = 28\%$.

Table 3: Validation Contingency Table			
	$\wedge(M_{ij} \leq M_{cj})$ $C_i \leq 38 \wedge S_i \leq 26$	$\vee(M_{ij} > M_{cj})$ $C_i > 38 \vee S_i > 26$	
High Quality $F_i \leq F_c$ <i>drcount=0</i>	$C_{11}=30$	$C_{12}=27$ Type 2	$n_1=57$
Low Quality $F_i > F_c$ <i>drcounFO</i>	$C_{21}=1$ Type 1	$C_{22}=42$	$n_2=43$
	$N_1=31$ RF=1, RFM=1	$N_2=69$	$n=100$ TF=192
	ACCEPT	REJECT	

Table 4: Application Contingency Table			
	$\wedge(M_{ij} \leq M_{cj})$ $C_i \leq 38 \wedge S_i \leq 26$	$\vee(M_{ij} > M_{cj})$ $C_i > 38 \vee S_i > 26$	
High Quality ?	?	Type 2 ?	?
Low Quality ?	Type 1 ?	?	?
	$N_1=40$	$N_2=60$	$n=100$
	ACCEPT	REJECT	

Application Criteria

It is insufficient to just validate with respect to statistical criteria. In the final analysis, it is the performance of the metrics in the application context that counts. Therefore, validate metrics with respect to the application criteria: Quality and Inspection [SCH95 1, SCH952, SCH94].

Quality

First estimate the ability of the metrics to correctly classify quality, given that the quality is known to be low:

$$\text{LQC: Proportion of low quality (i.e., } drcount > 0) \text{ software correctly classified} = C_{22}/n_2 \quad (7)$$

For the example, $\text{LQC} = (42/43) * 100 = 97.7\%$.

As part of the application validation, estimate the Discriminative Power of M_{c_j} by summing quality factor in the *ACCEPT* column in Table 3 to produce Remaining Factor RF (e.g., remaining *drcount*), given by equation (8). This is the sum of F_i not caught by inspection because $(F_i > F_c) \wedge (M_{i1} \leq M_{c1}) \dots \wedge (M_{ij} \leq M_{cj}) \dots \wedge (M_{im} \leq M_{cm})$ for these modules.

$$\text{RF} = \sum_{i=1}^n F_i \text{ FOR } (F_i > F_c) \wedge (M_{i1} \leq M_{c1}) \dots \wedge (M_{ij} \leq M_{cj}) \dots \wedge (M_{im} \leq M_{cm}) \quad (8)$$

for $j=1, \dots, m$.

Estimate the proportion of RF by equation (9), where TF is the total F_i prior to inspection.

$$\text{RFP} = \text{RF}/\text{TF} \quad (9)$$

and

$$\text{TF} = \sum_{i=1}^n F_i$$

For the example, from Table 3 there is a one DR on one module that is incorrectly classified (i.e., $\text{RF} = 1$). The total number of DRs for the 100 modules is 192. Therefore, $\text{RFP} = (1/192) * 100 = .52\%$.

Estimate the density of RF by equation (10).

$$\text{RFD} = \text{RF}/n \quad (10)$$

For the example, $\text{RFD} = 1/100 = .01 \text{ } drcount/module$.

In addition estimate the count of modules remaining that have $F_i > F_c$. The proportion remaining RMP is given by equation (11). Note that $\text{RMP} = P_1$ (proportion of Type I misclassifications) when $F_c = 0$ (i.e., the only modules with $F_i > 0$ will be in the $C_{..}$ cell); see Table 3.

$$RMP = RFM/n, \quad (11)$$

where RFM is given by:

$$RFM = \text{COUNT}_{i=1}^n \text{ FOR } ((F_i > 0) \wedge (M_{i1} \leq M_{c1}) \wedge (M_{ij} \leq M_{cj}) \wedge (M_{im} \leq M_{cm}))$$

For the example, there is one accepted module with one DR, so $RMP = (1/100) * 100 = 1\%$.

Inspection

Inspection is one of the costs of high quality. You would be interested in weighing inspection requirements against the quality that is achieved for various values of M_{cj} . Estimate inspection requirements by noting that all modules with $M_{ij} > M_{cj}$ must be inspected; this is the count $C_{12} + C_{22}$. Thus the proportion of modules that must be inspected is given by:

$$I = (C_{12} + C_{22})/n \quad (12)$$

For the example, $I = ((27 + 42)/100) * 100 = 69\%$.

Note that I is also equal to the proportion of modules rejected as a result of applying the critical values of metrics to the modules (see Tables 3 and 4) and that $1 - I$ is equal to the proportion of modules accepted. Thus, for the example, the proportion of modules accepted is 31%.

Part of this inspection is "wasted" because of *Type 2* (C_{12}) misclassifications (i.e., modules are inspected because they are incorrectly flagged). Estimate wasted inspection by using equation (13) for *Type 2* misclassifications:

$$RI = C_{22}/C_{12} \quad (13)$$

For the example, $RI = 42/27 = 1.56$.

Summary of Validation Results

The results of the validation example are summarized in Table 5. The properties of *dominance* and *concordance*, which were defined previously, were evident in these validation results and in other samples that have analyzed from this data. That is, a point is reached in adding metrics where *Discriminative Power* is not increased because: 1) the contribution of the dominant metrics in correctly classifying quality has already taken effect and 2) additional metrics essentially replicate the classification results of the dominant metrics -- the *concordance* effect. This result is due to the property of the BDF used as an OR function, which will cause a module to be rejected if only one of the module's metrics exceeds its critical value. These effects can only be observed if a marginal analysis is performed, where metrics are added to the set one-by-one and the calculations shown in Table 5 are made after each metric is added. For each added metric, its effect is evaluated with respect to both statistical and application criteria. In addition, a suitable stopping rule must be used to know when to stop adding metrics (see the next section).

Because, as mentioned earlier, the prominence of the metric *comments* was an unexpected result, an analysis was made in Table 5 of the best set of metrics excluding *comments*. Thus entering the metrics in the set according to the K-S rule, *statements*, *etal*, and *nodes* are shown in Table 5. Note that these three metrics are required to reach the same value of RFP as *comments* alone and that other measures of quality -- LQC and RMP -- are inferior. Thus *comments* alone is superior to the three metric set for this sample.

Table 5: Discriminative Power Validity Evaluation (Sample 1, n=100 modules)

Metric Set	Critical Values				Statistical Criteria				Application Criteria			
	C _c	S _c	E1 _c	N _c	P ₁ %	P ₂ %	χ ² _c	α _c for χ ² _c	LQC %	RFP %	RMP %	I %
C	38				2	21	33.2	8.4x10 ⁻⁹	95.3	1.56	2	62
C,S	38	26			1	27	26.7	2.4x10 ⁻⁷	97.7	0.52	1	69
C,S,E1	38	26	10		1	30	22.5	2.1x10 ⁻⁶	97.7	0.52	1	72
S,E1,N		26	10	11	3	22	28.6	9.1x10 ⁻⁸	93.0	1.56	3	62
K-S Distance	0.585	0.557	10.492	0.487								

Note: No improvement in the quality criteria (LQC, RFP, RMP) was achieved when additional metrics which are not shown in the table, were added.

Stage 4: Apply a Stopping Rule for Adding Metrics

One rule for stopping the addition of metrics to a BDF is to quit when RFP no longer decreases as metrics are added. This is the *maximum quality* rule. This rule is illustrated in Table 5. When a third metric, *etal*, is added, there is no decrease in RFP and RMP nor is there an increase in LQC. If it is important to strike a balance between quality and cost (i.e., between RFP and I), add metrics until the ratio of the relative change in RFP to the relative change in I is maximum, as given by the *Quality Inspection Ratio* in equation (14), where *I* refers to the previous RFP and I:

$$QIR = (\Delta RFP / RFP_i) / (\Delta I / I_i) \tag{14}$$

For the example, QIR(C→C,S) = ((1.52-1.56)/1.56) / ((69-62)/62) = 5.90.
 QIR(C,S→C,S,E1) = 0. Therefore, stop adding metrics after *statements* has been added. In this particular case, equation (14) produces the same metric set as the *maximum quality* rule.

COMPARISON OF VALIDATION WITH APPLICATION RESULTS

A comparison of the Validation Sample with the Application Samples with respect to statistical criteria is shown in Table 6. A comparison of the Validation Sample with the Application Samples with respect to application criteria is shown in Tables 7 and 8. As was mentioned, only metrics data is available when the validated metrics are applied. However, to have a basis for comparison with the validation results, the values

shown in Tables 6, 7, and 8 were computed *retrospectively*(i.e, after the application project was far enough along to be able to collect the quality factor data). RMP is not shown in these tables because it is equal to P_1 when $F_c=0$. The average relative error (validation-application/application) across eighteen comparisons between sample 1 versus samples 2, 3, 4 in Tables 6, 7, and 8 is 32.9% with a standard deviation of 31.3%.

Table 6: Statistical Criteria P1 and P2 for Metric Set: C,S Validation (Sample 1) vs. Application (Samples 2, 3, and 4), n=100 modules							
P1 : Percentage Type 1 Misclassification				P2: Percentage Type 2 Misclassification			
Sample 1	Sample 2	Sample 3	Sample 4	Sample 1	Sample 2	Sample 3	Sample 4
1.0	1.0	4.0	3.0	27.0	24.0	18.0	22.0

Table 7: Application Criteria LQC and RFP for Metric Set: C,S Validation (Sample 1) vs. Application (Samples 2,3, and 4), n=100 modules							
LQC: Percentage of low quality software (<i>drcount</i>>0) correctly classified				RFP: Percentage of quality factor (<i>drcount</i>) not caught by inspection			
Sample 1	Sample 2	Sample 3	Sample 4	Sample 1	Sample 2	Sample 3	Sample 4
97.7	97.3	91.1	93.2	.52	.62	3.01	1.50

Table 8: Application Criteria RFD and I for Metric Set: C,S Validation (Sample 1) vs. Application (Samples 2,3, and 4), n=100 modules							
RFD: Density of quality factor (<i>drcount</i>/module) not caught by inspection				I: Percentage of modules inspected			
Sample 1	Sample 2	Sample 3	Sample 4	Sample 1	Sample 2	Sample 3	Sample 4
.01	.01	.05	.03	69	60	59	63

PREDICTABILITY VALIDATION MODEL

In addition to using the *Contingency Table* to classify the Validation Sample during validation and subsequently to apply the results during quality control to the Application Sample, various quality predictions, such as proportion of modules with zero and non-zero *drcount* and their confidence limits, are derived from the *Contingency Table*. This approach has the advantage of integrating quality control and quality prediction in one basic model. Thus the software developer is provided with validated metrics to both control and predict the quality of the Application Sample. The predictions are used by the developer to anticipate rather than react to quality problems. For example, the predictions provide indications of the quality of the software that would be delivered to the customer if remedial steps (e.g., inspection, testing) are not taken. In addition, the predictions provide indications of resource levels that are needed to achieve quality goals. For example, if the predicted quality of the software is lower than the specified quality, the

difference would be an indication of increased usage of personnel and computer time during inspection and testing, respectively. This is a new application of BDFs, developed by the author, used within the framework of a *Contingency Table*, to both control and predict software quality.

Two type of predictions are made: module counts and quality factor counts. Using Validation Sample 1, point estimates and confidence intervals are computed for each and compared with the actual values of Application Samples 2, 3, and 4. The normal approximation to the binomial distribution is used to compute the confidence limits of the proportions.

Module Counts

The proportion of modules that are either **not flagged** or **flagged** for inspection are estimated below and are shown in Table 9.

N_n : number of modules **not flagged** for inspection in the Validation Sample

N_n' : number of modules **flagged** for inspection in the Validation Sample

N_1 : number of modules **not flagged** for inspection in the Application Sample

N_1' : number of modules **flagged** for inspection in the Application Sample

o proportion of modules with $F_i > 0$ (e.g., *drcount* > 0 on module I) in the Validation Sample prior to inspection and correction of defects:

$$p_n = \frac{\sum_{i=1}^n \text{COUNT FOR } F_i > 0}{n} \quad (15)$$

where $\text{COUNT}(I) = \text{COUNT}(I-1) + 1$ FOR expression *true* and $\text{COUNT}(I) = \text{COUNT}(I-1)$, otherwise; $\text{COUNT}(0) = 0$.

o two-sided confidence limits of p_n , used as predicted limits of p_n' in the Application Sample:

$$CL_{p_n} = p_n \pm Z_{\alpha/2} \sqrt{\frac{p_n(1-p_n)}{n}} \quad (16)$$

o proportion of modules **not flagged** for inspection (i.e., contained in N_n) with $F_i > 0$ (e.g., *drcount* > 0 on module I) in the Validation Sample:

$$pN_1 = \text{RFM} / N_1 \quad (17)$$

o one-sided upper confidence limit of pN_1 , used as predicted limit of pN_1' in the Application Sample:

$$UL_{pN_1} = pN_1 + Z_{\alpha} \sqrt{\frac{(pN_1)(1-pN_1)}{N_1}} \quad (18)$$

- o proportion of modules **flagged** for inspection (i.e., contained in N_2) with $F_i > 0$ (e.g., $drcount > 0$ on module I) in the Validation Sample:

$$pN_2 = ((p_n)(n) - (RFM)) / N_2 \quad (19)$$

- o one-sided lower confidence limit of pN_2 , used as predicted limit of pN_2' in the Application Sample:

$$LLpN_2 = pN_2 - Z_{\alpha} \sqrt{\frac{(pN_2)(1-pN_2)}{N_2}} \quad (20)$$

Quality Factor Counts

The proportion of quality factor count (e.g., $drcount$) on modules that are either **not flagged** or **flagged** for inspection are estimated below and are shown in Table 9. In addition, point estimates of total quality factor count on the modules is estimated and shown in Table 10.

- o proportion of quality factor that occurs on modules **not flagged** for inspection (i.e., contained in N_1) in the Validation Sample:

$$d_1 = RF/TF \text{ (same as RFP if RFP is expressed as a proportion)} \quad (21)$$

- o one-sided upper confidence limit of d_1 , used as predicted limit of d_1' in the Application Sample

$$ULd_1 = d_1 + Z_{\alpha} \sqrt{\frac{(d_1)(1-d_1)}{TF}} \quad (22)$$

- o proportion of quality factor that occurs on modules **flagged** for inspection (i.e., contained in N_2) in the Validation Sample:

$$d_2 = 1 - d_1 \quad (23)$$

- o one-sided lower confidence limit of d_2 , used as predicted limit of d_2' in the Application Sample:

$$LLd_2 = d_2 - Z_{\alpha} \sqrt{\frac{(d_2)(1-d_2)}{TF}} \quad (24)$$

- o expected quality factor count (e.g., $drcount$) that occurs on modules **not flagged** for inspection (i.e., contained in N_1') in the Application Sample (note that no knowledge of the *Application* quality factor is required to make this estimate):

$$D_1 = (RF/N_1)(N_1') \quad (25)$$

- o expected quality factor count (e.g., $drcount$) that occurs on modules **flagged** for inspection (i.e., contained in N_2') in the Application Sample (note that no knowledge of the *Application* quality factor is required to make this estimate):

$$D_2 = ((TF - RF) / N_2) (N_2') \quad (26)$$

Ten of the actual values out of the fifteen cases in Table 9 fall within the confidence limits. The average relative error across six comparisons between sample 1 versus samples 2, 3, 4 in Table 10 is 28.9% with a standard deviation of 30.7%.

Table 9: Validation Predictions (Sample vs. Application Actual Values (Samples 2, 3, and 4))					
	Point Estimates (Sample 1)	95% Limits (Sample 1)	Actual Values		
			Sample 2	Sample 3	Sample 4
p_n' : proportion of modules with <i>drcount</i> >0	43.0%	33.3%-52.7%	37.0%	45.0%	44.0%
pN_1' : proportion of modules not flagged for inspection with <i>drcount</i> >0	3.22%	LE 8.45%	2.50%	9.76%	8.11%
pN_2' : proportion of modules flagged for inspection with <i>drcount</i> >0	60.9%	GE 51.2%	60.0%	69.5%	65.1%
d_1' : proportion of <i>drcount</i> on modules not flagged for inspection	.52%	LE 1.38%	.62%	3.01%	1.50%
d_2' : proportion of <i>drcount</i> on modules flagged for inspection	99.5%	GE 98.6%	99.4%	97.0%	98.5%

Table 10: Validation Actual Values and Predictions (Sample 1) vs. Application Actual Values (Samples 2, 3, and 4)							
	Actual Sample 1	Predicted Sample 1	Actual Sample	Predicted Sample 1	Actual Sample 3	Predicted Sample 1	Actual Sample 4
D_1' : expected <i>drcount</i> on modules not flagged for inspection	1	1.29	1	1.32	5	1.19	3
D_2' : expected <i>drcount</i> on modules flagged for inspection	191	166.1	160	163.3	161	174.4	197

INTEGRATION OF METRICS WITH QUALITY AND RELIABILITY

For metrics to be used as leading indicators of quality and reliability, you must show that the modules that cause failures have both a significant number of DRs written against them and that the modules' key metrics have values that exceed the critical values. To make a tie with the *Discriminative Power Validation Model*, an example of this process is shown in Table 11, listing the DR count and values for the thirteen metrics of the five failures experienced by the *Space Shuttle* Operational Increment M (a series of builds to meet mission requirements). Because software failures are rare on the *Space Shuttle*, only five failures are shown. But this amount of data is sufficient to demonstrate the concept. In the table there are multiple DRs written against the failed modules and the *comments* and *statements* values -- this is the optimal set of metrics derived in Table 5 -- exceed their critical values by significant amounts. Also there are only three cases (bolded) where any of the thirteen metrics' values do not exceed their critical value. Also, for comparison purposes, the mean and median of the metrics' values for the 1397 modules that comprise the *Space Shuttle* Operational Increment M are shown. In general, the metrics values of the failed modules are far in excess of the mean and median.

Also, to make a tie with the *Predictability Validation Model*, the proportion of the 1397 modules that have DRs was calculated as 41.2%, which is within the confidence limits predicted in the first row of Table 9.

Table 11: Integration of Metrics with Quality and Reliability

Failed Module DR and Metrics Counts, Operational Increment M								
Metric	Critical Value	Module 1	Module 2	Module 3	Module 4	Module 5	Population Module Mean	Population Module Median
DRs	0	13	9	5	6	2	1.8	0
comments	38	448	204	79	434	68	134.6	64
eta1	10	39	37	38	30	23	16.7	13
statements	26	366	261	103	563	36	70.2	23
nodes	11	70	10	21	137	16	28.4	7
edges	10	90	11	27	181	18	36.2	6
loc	19	484	326	140	485	83	132.3	61
maxpath	8	80	9	25	94	12	24.5	6
paths	1	10000	4	224	10000	6	1587.2	2
avepath	4	67.7	8.00	19.4	78.5	11.0	20.7	5
eta2	33	344	162	126	166	52	81.3	46
$\eta 1$	26	1712	1089	386	2526	458	480.0	171
$\eta 2$	21	1026	608	292	1358	216	307.7	106
cycles	0	3	0	1	2	0	0.93	0

CONCLUSIONS

It is important when validating and applying metrics to consider both statistical and application criteria and to measure the marginal contribution of each metric in satisfying these criteria. When this approach is used, a point is reached where adding metrics makes no contribution to improving quality and the cost of using additional metrics increases. This phenomenon is due to the metric classification properties of *dominance* and *concordance*. The ratio of the relative improvement in quality to the relative increase in cost provides a good stopping rule for adding metrics. The Boolean discriminant function (BDF) is a new type of discriminant for classifying software quality to support an *integrated* approach to control and prediction in one model, and the application of Kolmogorov-Smirnov (K-S) distance is a new way to determine a metric's critical value. It was shown that the metrics *comments* and *statements*, when combined in a single BDF, were better indicators (i.e., $\leq 3\%$ error in classifying low quality modules) of the quality of the *Space Shuttle* software than complexity metrics.

References

- [CON713] W. J. Conover, Practical Nonparametric Statistics, John Wiley & Sons, Inc., 1971.
- [IEE93] Standard for a Software Quality Metrics Methodology, IEEE Std 1061-1992, March 12, 1993.
- [IEE90] IEEE Glossary of Software Engineering Terminology, 610.12, 1990.
- [SCH95 13] Norman F. Schneidewind, "Software Metrics Validation: Space Shuttle Flight Software Example", Annals of Software Engineering, J. C. Baltzer AG, Science Publishers, 1995.
- [SCH952] Norman F. Schneidewind, "Controlling and predicting the quality of space shuttle software using metrics", Software Quality Journal **4**, 49-68, (1995), Chapman & Hall.
- [SCH94] Norman F. Schneidewind, "Validating Metrics for Controlling and Predicting the Quality of Space Shuttle Flight Software", IEEE Computer, Vol. 27, No. **8**, August, 1994, pp. 50-57.
- [SCH92] Norman F. Schneidewind, "Methodology for Validating Software Metrics", IEEE Transactions on Software Engineering, Vol. 18, No. 5, May 1992, pp. 410-422.

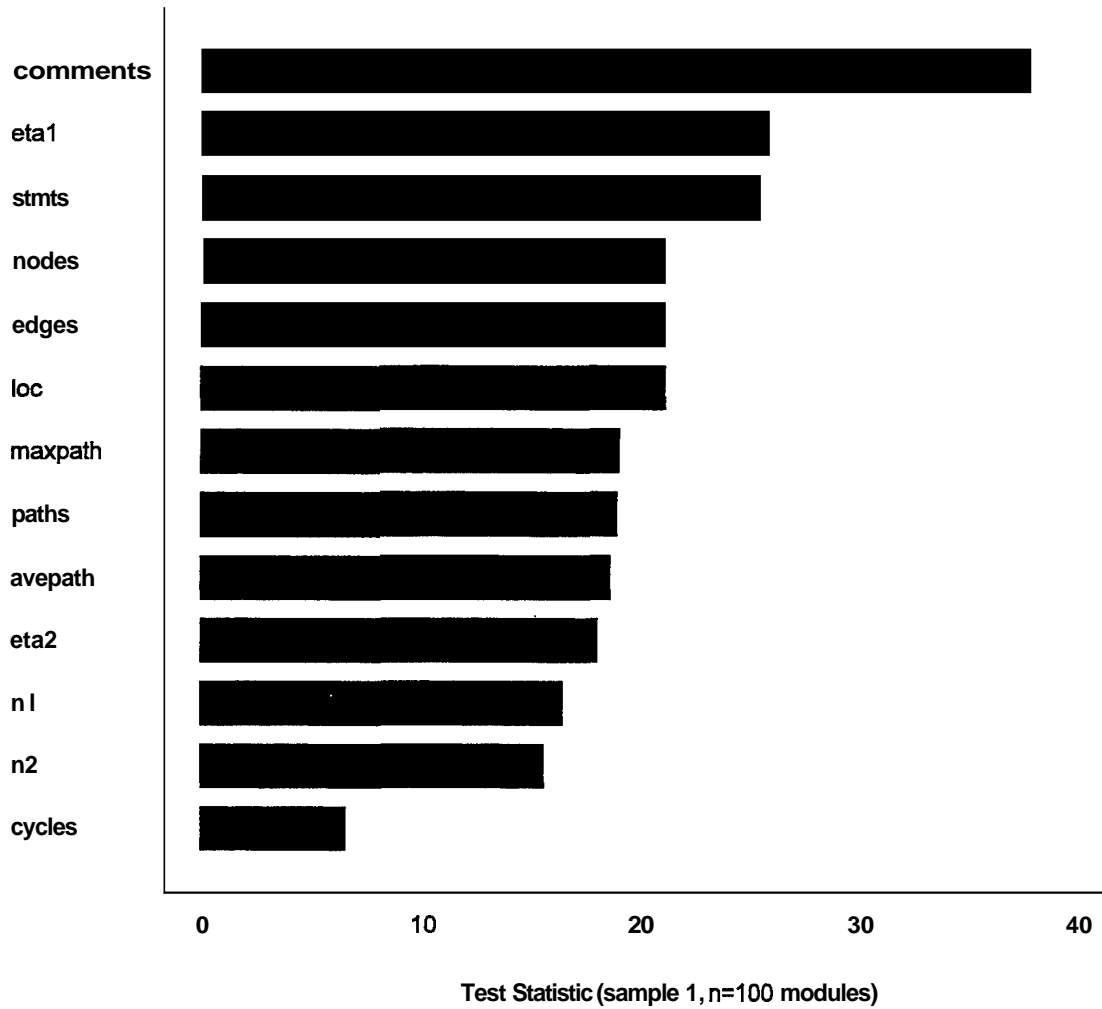


Figure 1. Kruskal-Wallis Analysis of Metrics

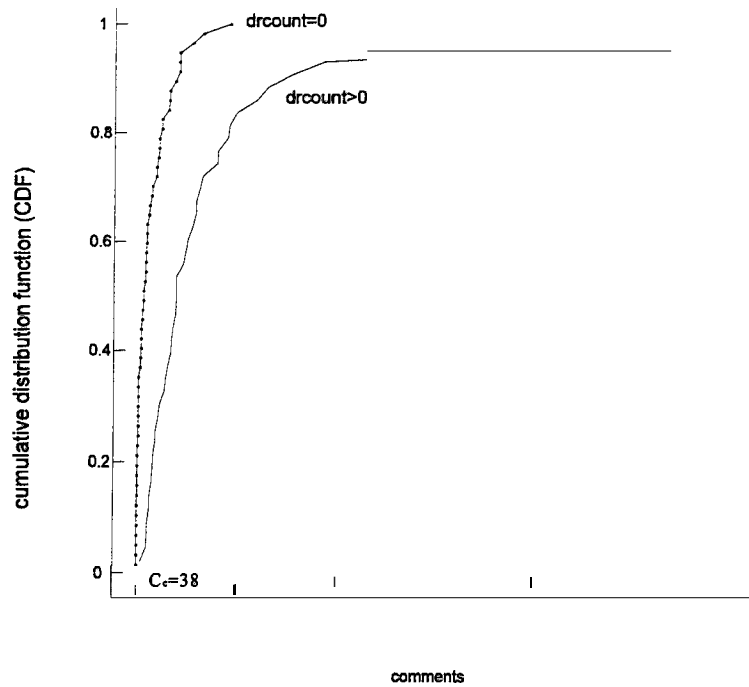


Figure 2. K-S Test: comments CDF (sample 1, n=100 modules)

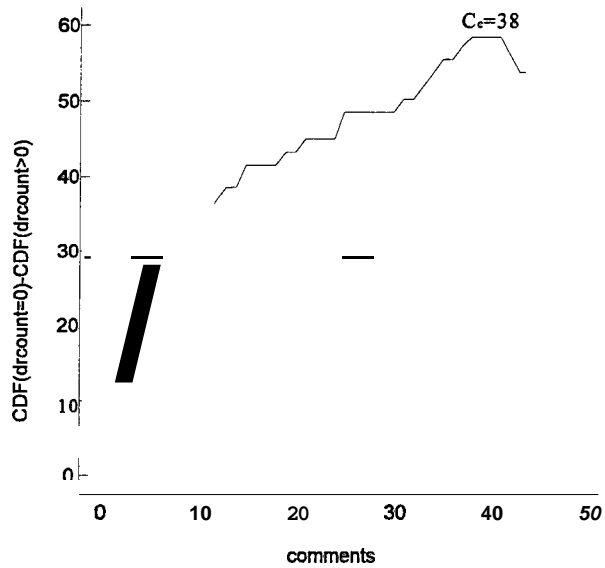


Figure 3. Difference in CDFs for comments (sample 1, n=100 modules)

**APPENDIX A. SAMPLE STATGRAPHICS METRICS ANALYSIS SESSION
DISCRIMINATIVE POWER VALIDATION MODEL**

Comments are in parentheses to distinguish them from Statgraphics input and output. Equation, figure, and table numbers refer to the text.

(Stage 1 of the metrics analysis: Identify Candidate Metrics)

Kruskal-Wallis One-way Analysis by Ranks

.....

Data: comments SELECT sample EQ 1

Level codes: drcount EQ 0

(This is an example of the setup menu for Stage 1 of the metrics analysis. The menu was entered from the **Analysis of Variance** and **Krusal-Wallis One-way Analysis by Ranks**. The metric *comments* and the random *Validation Sample I* of 100 modules have been specified. The quality factor *drcount* is used to separate the modules into two classes: drcount EQ 0 and drcount GT 0.)

Kruskal-Wallis analysis of comments SELECT sample EQ 1 by drcount EQ 0

Level	Sample Size	Average Rank
0	43	71.0581
1	57	34.9912

Test statistic = **37.913** Significance level = **7.39689E-10**

.....

(The 57 modules with drcount EQ 0 (Level 1) have an average rank of 34. The 43 modules with drcount GT 0 (Level 0) have an average rank of 71. The K-W test statistic of 37 is highly significant as indicated by the significance level, which is close to 0. These values are entered in Table 2 and used in Figure 1. This process is repeated for the remaining 12 metrics.)

(Stage 2: Compute Critical Values)

Percentiles

Data vector: comments SELECT ((drcount EQ 0) AND (sample EQ 1))

Percentages comments

12.2807 = 0

17.5439 = 1

21.0526 = 2

22.807 = 3

22.807 = 4

35.0877 = 5

35.0877 = **6**

35.0877 = 7

35.0877 = 8

35.0877 = 9

35.0877 = 10

35.0877 = 11

36.8421 = 12

38.5965 = 13

38.5965 = 14

(This is an example of Stage 2 of the metrics analysis. The menu was entered from **Descriptive Methods** and **Percentiles**. The objective is to form and save a CDF of *comments* for modules with *drcount* EQ 0. The left column is the CDF in % for the specified values of *comments* in the right column. This part of the CDF is saved in a file for future computation of C_c . These values are used in Figure 2. Note that the editor was used to replace the original title *Percentiles* with *comments* in the right column.)

Enter variable in which to save the percentages:

File: lowlevel Variable: commcdf01

(The first part of the CDF for *comments* is saved in the file *lowlevel* as variable *commcdf01*. The process is repeated for the second part (*comments*=15,29) and third part (*comments*=30,44) except these parts are appended to the original file. Three parts are sufficient to compute C_c .)

Percentiles

Data vector: comments SELECT ((drcount GT 0) AND (sample EQ 1))

Percentages	comments
0	= 0
0	= 1
0	= 2
0	= 3
0	= 4
0	= 5
0	= 6
0	= 7
0	= 8
0	= 9
0	= 10
0	= 11
0	= 12
0	= 13
0	= 14

(The objective is to form and save a CDF of *comments* for modules with *drcount* GT 0. The left column is the CDF in % for the specified values of *comments* in the right column. This part of the CDF is saved in a file for future computation of C,. These values are used in Figure 2. Note that the editor was used to replace the original title *Percentiles* with *comments* in the right column.)

(The following is a continuation of Stage 2 designed to find the values of *comments* with the maximum difference between the CDFs for *drcount* EQ 0 and *drcount* GT 0. **This** process is the implementation of equation (2). The process starts with *comments* because it was the metric with the largest value of the K-W statistic in Stage 1. When an equation is used for the first time, it is displayed by typing its name. When an equation is computer, its name is preceded by the reserved word **EVAL**). Inputs have a colon prompt and outputs have no colon.)

: maxcdfdiff (Equations for computing the maximum difference in CDFs)

MAX (EVAL (cdfdiff))

: cdfdiff

(ABS(m1-m2))/100

: m1 GETS commcdf01 (The CDF for *drcount* EQ 0 is loaded)

: m2 GETS commcdf1 1 (The CDF for *drcount* GT 0 is loaded)

: EVAL maxcdfdiff (The maximum difference is computed and entered in Table 2)

0.585067

: (commcdf01/100)-(commcdf11/100)

0.122807 0.175439 0.210526 0.22807 0.22807 0.350877 0.350877 0.350877

0.350877 0.350877 0.350877 0.350877 0.368421 0.385965 0.385965

0.415341 0.415341 0.415341 0.415341 0.432885 0.432885 0.450428

0.450428 0.450428 0.450428 0.485516 0.485516 0.485516 0.485516

0.485516 0.485516 0.50306 0.50306 0.520604 0.538148 0.555692 0.555692

0.573235 **0.585067** 0.585067 0.585067 0.585067 0.561812 0.538556

0.538556

(The difference in the CDFs is computed. These data are used in Figure 3. The value of $C_c=38$ is determined by counting from upper left, top to bottom, left to right, remembering that the first value is $C=0$. The critical value of *comments* is entered in Table 2. This process is repeated for as many metrics as are required in the Stage 3 analysis -- see below -- taking the metrics in the order of their K-W statistic.)

(Stage 3: Perform Contingency Table Analysis)

(The following are computations dealing with Stage 3 of the metrics analysis. Equation (3) is implemented using a formula which can handle up to four metrics. There is also a version of this formula which can handle up to six metrics. This example shows a BDF consisting of *comments* and *stmts*, and their critical values. Because the computations for C11, C12, C21, and C22 require that a metric be specified for each of the four or possible six metrics, *stmts* is used in the computation three times. Because the and OR Boolean operations are not affected by the repetition of metrics, the correct result is achieved. Various quantities -- both statistical and quality -- are computed as part of the validation process).

: Ss GETS 1 (Load ID of random sample *I* of 100 modules)

(The following are *Validation Sample I* computations for Table 3)

:C11 (Module count for C,, of equation (3) and Table 3)

SUM((drcount LE Dc) AND (M1 LE M1c) AND (M2 LE M2c) AND (M3 LE M3c) AND (M4 LE M4c)
AND (sample EQ Ss))

:Dc GETS 0 (Load critical value of *drcount*)

:M1 GETS *comments* (Load metric vector *comments*)

:M2 GETS *stmts* (Load metric vector *stmts* three times)

:M3 GETS *stmts*

:M4 GETS *stmts*

:M1c GETS 38 (Load critical value of *comments*)

:M2c GETS 26 (Load critical value of *stmts* three times)

:M3c GETS 26

:M4c GETS 26

:EVAL C11 (Compute module count for C,,. Enter in Table 3)

: c 12 (Module count for C_{12} , of equation (3) and Table 3)
SUM ((drcount LE Dc) AND ((M1 GT M1c) OR (M2 GT M2c) OR (M3 GT M3c) OR (M4 GT M4c))
AND (sample EQ Ss))

: EVAL C12 (Compute module count for C_{12} . Enter in Table 3)

27

: c 21 (Module count for C_{21} of equation (3) and Table 3)
SUM ((drcount GT Dc) AND (M1 LE M1c) AND (M2 LE M2c) AND (M3 LE M3c) AND (M4 LE M4c))
AND (sample EQ Ss))

: EVAL C21 (Compute module count for C_{21} . Enter in Table 3)

1

: c 22 (Module count for C_{22} , of equation (3) and Table 3)
SUM ((drcount GT Dc) AND ((M1 GT M1c) OR (M2 GT M2c) OR (M3 GT M3c) OR (M4 GT M4c))
AND (sample EQ Ss))

: EVAL C22 (Compute module count for C_{22} . Enter in Table 3)

42

: N1 (Count of accepted modules in Table 3)
SUM ((M1 LE M1c) AND (M2 LE M2c) AND (M3 LE M3c) AND (M4 LE M4c) AND (sample EQ Ss))
: EVAL N1 (Compute accepted module count. Enter in Table 3)

31

: N2 (Count of rejected modules in Table 3)
SUM (((M1 GT M1c) OR (M2 GT M2c) OR (M3 GT M3c) OR (M4 GT M4c)) AND (sample EQ Ss))
: EVAL N2 (Compute rejected module count. Enter in Table 3)

69

:n1 (Count of high quality modules in Table 3)
(EVAL C11)+(EVAL C12)
:EVAL n1 (Compute count of high quality modules. Enter in Table 3)

57

:n2 (Count of low quality modules in Table 3)
(EVAL C21)+(EVAL C22)
:EVAL n2 (Compute count of low quality modules. Enter in Table 3)

43

:TF (Total *drcount* of equation (9) and Table 3)
SUM (drcount SELECT sample EQ Ss)
:EVAL TF (Compute total *drcount*. Enter in Table 3)

192

:RF (Remaining *drcount* of equation (9) and Table 3)
SUM (drcount SELECT ((M1 LE M1c) AND (M2 LE M2c) AND (M3 LE M3c) AND (M4 LE M4c) AND
(drcount GT Dc) AND (sample EQ Ss)))
:EVAL RF (Compute remaining *drcount*. Enter in Table 3)

1

:RFM (Module count with remaining *drcount* of equation (11) and Table 3)
SUM ((drcount GT Dc) AND (M1 LE M1c) AND (M2 LE M2c) AND (M3 LE M3c) AND (M4 LE M4c)
AND (sample EQ Ss))
:EVAL RFM (Compute module count with remaining *drcount*. Enter in Table 3)

1

:Ss GETS 2 (Load ID of random sample 2 of 100 modules)

(The following are *Application Sample 2* computations for Table 4)

: EVAL N1 (Compute accepted module count. Enter in Table 4)

40

: EVAL N2 (Compute rejected module count. Enter in Table 4)

60

: Ss GETS 1 (Load ID of random sample *I* of 100 modules)

(The following are *Validation Sample 1* computations for Tables 5-8)

: P1 (Proportion of Type 1 misclassifications of equation (4) and Tables 5 and 6)

$((\text{EVAL C21})/n) * 100$

: EVAL P1 (Estimate proportion of Type 1 misclassifications of. Enter in Tables 5 and 6)

1 %

: P2 (Proportion of Type 2 misclassifications of equation (5) and Tables 5 and 6)

$((\text{EVAL C12})/n) * 100$

: EVAL P2

27 % (Estimate proportion of Type 2 misclassifications of. Enter in Tables 5 and 6)

: P12 (Proportion of Type 1+Type 2 misclassifications of equation (6))

$((\text{EVAL C12})+(\text{EVAL C21}))/n) * 100$

: EVAL P12 (Estimate Proportion of Type 1+Type 2 misclassifications)

28 %

: LQC (Proportion of low quality software correctly classified of equation (7) and Tables 5 and 7)

$((\text{EVAL C22})/(\text{EVAL n2})) * 100$

: EVAL LQC (Estimate proportion of low quality software correctly classified. Enter in Tables 5 and 7)

97.6744 %

:RFP (Proportion of RF of equation (9) and Tables 5 and 7)
 $((\text{EVAL RF})/(\text{EVAL TF})) * 100$

: EVAL RFP (Estimate proportion of RF. Enter in Tables 5 and 7)
0.520833 %

:RFD (Density of RF of equation (10) and Table 8)
 $(\text{EVAL RF})/n$

: EVAL RFD (Estimate density of RF. Enter in Table 8)
0.01 *dr*count/module

:RMP (Proportion of RFM of equation (11))
 $((\text{EVAL RFM})/n) * 100$

: EVAL RMP (Estimate proportion of RFM)
1 %

: I (Proportion of modules that must be inspected of equation (12) and Tables 5 and 8)
 $((\text{EVAL C12})+(\text{EVAL C22}))/n * 100$

: EVAL I (Estimate proportion of modules that must be inspected. Enter in Tables 5 and 8)
69 %

:RI (Wasted inspection of equation (13))
 $(\text{EVAL C22})/(\text{EVAL C12})$

: EVAL RI (Estimate wasted inspection)
1.55556

Contingency Tables

Table matrix: 2 2 RESHAPE 30 27 1 42 (Enter the values C_{11} , C_{12} , C_{21} , and C_{22})

(This is an example of the setup menu using Contingency Tables for *Validation Sample 1* entered from **Categorical Data Analysis** and **Contingency Tables**. The operation "2 2 RESHAPE" converts the vector of cell values to a 2x2 matrix).

(Compute χ^2_c and \mathbf{a} , for χ^2_c of Table 5)

Summary Statistics for Contingency Tables

Chi-square	D.F.	Significance
------------	------	--------------

26.6941	1	2.383483-7 with Yates correction (Enter in Table 5)
----------------	---	--

: Ss GETS 2 (Load ID of random sample 2 of 100 modules)

(The following are *Application Sample 2* computations for Tables 6-8)

: EVAL P1 (Estimate proportion of Type 1 misclassifications of. Enter in Table 6)

1 *Y_o*

: EVAL P2 (Estimate proportion of Type 2 misclassifications of. Enter in Table 6)

24 *Y_o*

: EVAL LQC (Estimate proportion of low quality software correctly classified. Enter in Table 7)

97.2973 *Y_o*

: EVAL RFP (Estimate proportion of RF. Enter in Table 7)

0.621118 *Y_o*

: EVAL RFD (Estimate density of RF. Enter in Table 8)

0.01 *drcount/module*

: EVAL I (Estimate proportion of modules that must be inspected. Enter in Table 8)

60 %

PREDICTABILITY VALIDATION MODEL

:pn (Proportion of modules with $drcount > 0$ of equation (15) and Table 9)

$(SUM((drcount > 0) AND (sample EQ Ss)))/n$

: Ss GETS 1 (Load ID of random sample I of 100 modules)

(The following are *Validation Sample 1* point estimates used to compute confidence limits for

Application Sample 2 in Table 9)

: EVAL pn (Compute proportion of modules with $drcount > 0$. Enter in Table 9)

0.43

: CLpn (Two-sided confidence limits of p_n , of equation (16) and Table 9)

$((EVAL pn) + (Z * (SQRT (((EVAL pn) * (1 - (EVAL pn))) / n))))$

: Z GETS 1.96 (Set computation for 95% upper confidence limit)

: EVAL CLpn (Compute upper confidence limit of p_n . Enter in Table 9)

: Z GETS -1.96 (Set computation for 95% lower confidence limit)

: EVAL CLpn (Compute lower confidence limit of p_n . Enter in Table 9)

0.332965

: pN1 (Proportion of modules **not flagged** for inspection of equation (17) and Table 9)

$(EVAL RFM) / (EVAL N1)$

: EVAL pN1 (Compute proportion of modules **not flagged** for inspection. Enter in Table 9)

0.0322581

: ULpN1 (Upper Confidence limit of p_{N1} of equation (18) and Table 9)

$((EVAL pN1) + (Z * (SQRT (((EVAL pN1) * (1 - (EVAL pN1))) / (EVAL N1))))$

: Z GETS 1.6449 (Set computation for 95% confidence limit)

: EVAL UL_{pN1} (Compute upper confidence limit of p_{N1}. Enter in Table 9)

0.0844565

: p_{N2} (Proportion of modules **flagged** for inspection of equation (19) and Table 9)

$((n*(EVAL\ p_n))-(EVAL\ RFM))/(EVAL\ N_2)$

: EVAL p_{N2} (Compute proportion of modules **flagged** for inspection. Enter in Table 9)

0.608696

: LL_{pN2} (Lower confidence limit of p_{N2} of equation (20) and Table 9)

$((EVAL\ p_{N2})-(Z*(SQRT\ (((EVAL\ p_{N2})*(1-(EVAL\ p_{N2}))))/(EVAL\ N_2))))$

: EVAL LL_{pN2} (Compute lower confidence limit of p_{N2}. Enter in Table 9)

0.512052

: d₁ (Proportion of *drcount* that occurs on modules not flagged for inspection of equation (21) and Table 9)

$(EVAL\ RF)/(EVAL\ TF)$

: EVAL d₁ (Compute proportion of *drcount* that occurs on modules not flagged for inspection. Enter in Table 9)

0.00520833

: UL_{d1} (Upper confidence limit of d, of equation (22) and Table 9)

$((EVAL\ d_1)+(Z*(SQRT\ (((EVAL.\ d_1)*(EVAL\ d_2))/(EVAL\ TF))))$

: EVAL UL_{d1} (Compute upper confidence limit of d, . Enter in Table 9)

0.0137532

: d₂ (Proportion of *drcount* that occurs on modules flagged for inspection of equation (23) and Table 9)

$(1-EVAL\ (d_1))$

: EVAL d₂ (Compute proportion of *drcount* that occurs on modules flagged for inspection. Enter in Table 9)

0.994792

:LLd2 (Lower confidence limit of d_2 of equation (24) and Table 9)

$$((\text{EVAL } d_2) - (Z * (\text{SQRT } (((\text{EVAL } d_1) * (\text{EVAL } d_2)) / (\text{EVAL } TF))))))$$

: EVAL LLd2 (Compute lower confidence limit of d_2 . Enter in Table 9)

0.986247

(The following are *Validation Sample 1* expected value computations for Table 10)

:D1 (Expected *drcount* that occurs on modules **not flagged** for inspection of equation (25) and Table 10)

$$((\text{EVAL } RF) / (\text{EVAL } N1)) * N1a$$

:N1a GETS 31 (Computation of D_1 is for *Validation Sample 1*. Thus, $N1a = N1 = "31"$, computed in an earlier step)

: EVAL D1 (Compute expected *drcount* that occurs on modules **not flagged** for inspection. Enter in Table 10)

1

:D2 (Expected *drcount* that occurs on modules **flagged** for inspection of equation (26) and Table 10)

$$(((\text{EVAL } TF) - (\text{EVAL } RF)) / (\text{EVAL } N2)) * N2a$$

:N2a GETS 69 (Computation of D_2 is for *Validation Sample 1*. Thus, $N2a = N2 = "69"$, computed in an earlier step)

: EVAL D2 (Compute expected *drcount* that occurs on modules **flagged** for inspection. Enter in Table 10)

191

:Ss GETS 2 (Load ID of random sample 2 of 100 modules)

(The following are *Application Sample 2* proportions actual values computations for Table 9)

: EVAL pn (Compute proportion of modules with *drcount* > 0. Enter in Table 9)

0.37

: EVAL pN1 (Compute proportion of modules **not flagged** for inspection. Enter in Table 9)

0.025

: EVAL pN2 (Compute proportion of modules **flagged** for inspection. Enter in Table 9)

0.6

: EVAL d1 (Compute proportion of *drcount* that occurs on modules **not flagged** for inspection). Enter in Table 9)

0.00621118

: EVAL d2 (Compute proportion of *drcount* that occurs on modules **flagged** for inspection). Enter in Table 9)

0.993789

: Ss GETS 1 (Load ID of random sample *I* of 100 modules)

(The following are *Application Sample 2* expected value computations for Table 10. However, values from *Validation Sample 1* are used except for N1a and N2a)

: N1a GETS 40 (Computation of D_1 is for *Application Sample 2*. Thus, $N1a=N1 = "40"$, computed in an earlier step)

: EVAL D1 (Compute expected *drcount* that occurs on modules **not flagged** for inspection. Enter in Table 10)

1.29032

: N2a GETS 60 (Computation of D_2 is for *Application Sample 2*. Thus, $N2a=N2= "60"$, computed in an earlier step)

: EVAL D2 (Compute expected *drcount* that occurs on modules **flagged** for inspection. Enter in Table 10)

166.087

DISTRIBUTION LIST

<u>Agency</u>	<u>No. of Copies</u>
Defense Technical Information Center 8725 John J. King Rd., STE 0944 Ft. Belvoir, VA 22314	2
Dudley Knox Library, Code 013 Naval Postgraduate School Monterey, CA 93943	2
Office of Research Administration, Code 91 Naval Postgraduate School Monterey, CA 93943	1
Department of Systems Management Library, Code SM/Eb Naval Postgraduate School 555 Dyer Rd Rm 239 Bldg. 330 Monterey, CA 93943	1
Commanding Officer Marine Corps Tactical Systems Support Activity Box 555171 Camp Pendleton. CA 92055-5171	1
Capt. Kenneth Warburton Marine Corps Tactical Systems Support Activity Box 555171 Bldg. 31345 Camp Pendleton, CA 92055-5171	5
Dr. Norman F. Schneidewind Naval Postgraduate School Code SM/Ss Monterey, CA 93943	10