Methodology For Validating Software Metrics

Norman F. Schneidewind, *Senior Member, IEEE*

Abstract— **We propose a comprehensive metrics validation methodology that has six validity criteria, which support the quality functions assessment, control, and prediction, where quality functions are activities conducted by software organizations for the purpose of achieving project quality goals. Six criteria are defined and illustrated: association, consistency, discriminative power, tracking, predictability, and repeatability. We show that nonparametric statistical methods such as contingency tables play an important role in evaluating metrics against the validity criteria. Examples emphasizing the discriminative power validity criterion are presented. A metrics validation process is defined that integrates quality factors, metrics, and quality functions.**

Index Terms—**Metrics validation methodology, metrics validation process, nonparametric statistical methods, quality functions, validity criteria.**

I. Introduction

WE believe that software metrics should be treated as part of an engineering discipline—metrics should be evaluated (validated) to determine whether they measure what they purport to measure prior to using them. Furthermore, if metrics are to be of greatest utility, the validation should be performed in terms of the quality functions (quality assessment, control, and prediction) that the metrics are to support.

We propose and illustrate a validation methodology whose adoption, we believe, would provide a rational basis for using metrics. This is a comprehensive metrics methodology that builds on the work of others: these have been validation analyses performed on specific metrics or metric systems for the purpose of satisfying specific research goals. Among these validations are the following: 1) function points as a predictor of work hours across different development sites and sets of data [1]; 2) reliability of metrics data reported by programmers [3]; 3) Halstead operator count for Pascal programs [10]; 4) metric-based classification trees [16]; and 5) evaluation of metrics against syntactic complexity properties [17].

Our approach to validation has the following characteristics: (i) The methodology is general and not specific to particular metrics or research objectives. (ii) It is developed from the point of view of the metric user (rather than the researcher), who has requirements for assessing, controlling, and predicting quality. To illustrate the difference in viewpoint, we can make an analogy with the automobile industry: the manufacturer has an interest in brake lining thickness as it relates to stopping distance, but from the driver's perspective, the only meaningful metric is stopping distance! (iii) It consists of six mathematically defined criteria, each of which is keyed to a quality function, so the user of metrics can understand how a characteristic of a metric, as revealed by validation tests, can be applied to measure software quality. (iv) The six criteria are: association, consistency, discriminative power, tracking, predictability, and repeatability. (v) It recognizes that a given metric can have multiple uses (e.g., assess, control, and predict quality) and that a given metric can be valid for one use and invalid for another use. (vi) It defines a metrics validation process that integrates quality factors, metrics, and functions.

This paper is organized as follows. First, in Section II a framework is established which pulls together the concepts and definitions of quality factor, quality metric, validated metric, quality function, validity criteria, and a metrics validation process. These concepts and definitions are integrated by the use of a metrics validation process chart. In this section we show how validity criteria support quality functions. Next, in Section III we indicate why nonparametric statistical methods are applicable to and compatible with the validity criteria. This is followed in Section IV by an example of metrics validation, using the discriminative power validity criterion. Lastly, in Section V some comments are made about future research directions.

II. Framework

The framework of our metrics methodology consists of the following elements, which are keyed to Fig. 1: the quality factor, quality metric, validated metric, quality functions, validity criteria, and metrics validation process. In Fig. 1 we use the notation (*Project, Time, Measurement*) to designate the project, time (e.g., life-cycle phase), and type of measurement (quality factor, quality metric). We use V to designate the project in which a metric is validated, and A to designate the project in which the metric is applied.

This diagram is interpreted as follows:

- The events and time progression of the validation project are depicted by the top horizontal line and arrow. This time line consists of Project 1 with metric M collection in Phase $T1$ (step 1); factor F collection in Phase $T2$ (step 2); and validation of M with respect to F in Phase $T2$ (step 3).
- The events and time progression of the application project are depicted by the bottom horizontal line and arrow. This project is later in chronological time than the validation project, but has the same phases $T1$ and $T2$. This time line consists of Project 2 with metric collection M' in Phase $T1$ (step 4); application of M' to assess, control, and predict quality in Phase $T1$ (step 5); collection of

Manuscript received December 19, 1990; revised January 15, 1992. Recommended by M. V. Zelkowitz. This work was supported by the Naval Surface Warfare Center, and by the Army Operational Test and Evaluation Center.

The author is with the Naval Postgraduate School, Monterey, CA 93943.

IEEE Log Number 9107760.

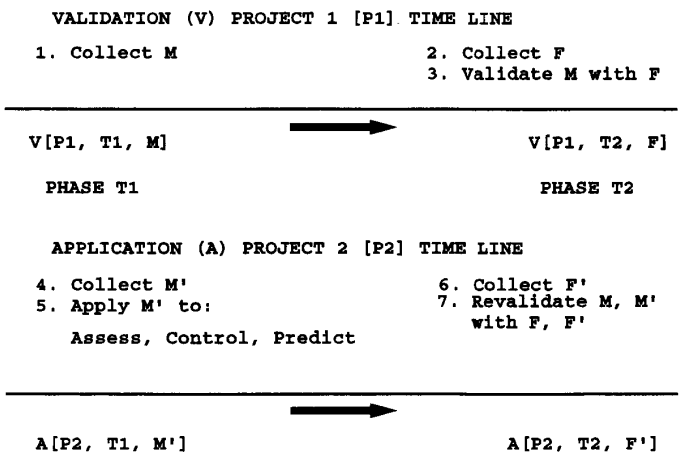

Fig. 1. Metrics validation process.

factor F' in Phase $T2$ (step 6); and revalidation of M and M' with respect to F and F' in Phase $T2$ (step 7).

- Metric M' is the same metric as M, but in general it has different values, since it is collected in a different project. The same statement applies to F' and F.

Each element is defined and described in greater detail in the following subsections.

A. Quality Factor

A quality factor F (hereafter referred to as "factor" or "F") is an attribute of software that contributes to its quality [13], where software quality is defined as the degree to which software possesses a desired combination of attributes [14]. For example, reliability (an attribute that contributes to quality) is a factor. A factor can have values such as the error counts $F_1, \ldots, F_n$ in a set of software components (i.e., an element of a software system, such as module, unit, data, or document [13]). We define F to be a type of metric that provides a direct measure of software quality [6]. This means that F is an intrinsic indicator of quality as perceived by the user, such as errors in the software that result in failures during operation. We denote F as the factor in V, and F' as the factor in A. F and F' are shown as collected at points 2 and 6, respectively, in Fig. 1.

B. Quality Metric

A quality metric M (hereafter called "metric" or "M") is a function (e.g., cyclomatic complexity $M = e - n + 2p$) whose inputs are software data (elementary software measurements, such as the number of edges e and number of nodes n 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) [15]. For example, if there are two components 1 and 2 with $M_1 = 3$ and $M_2 = 10$, this may indicate that the reliability of 1 may be greater than the reliability of 2. Whether this is the case depends upon whether M is a valid metric (see below). We define M to be an indirect measure of software quality [2], [6]. This means that M may be used as a substitute for F when F is not available, as is the case during the design phase. M is shown as collected at point 1 in Fig. 1.

It is important to recognize that in general there can be a many-to-many relationship between F and M. For expository purposes, we limit our examples to one-to-one or one (F) to many (M) relationships.

C. Validated Metric

A validated metric is one whose values have been shown to be statistically associated with corresponding factor values (e.g., $M_1, \ldots, M_n$ have been statistically associated with $F_1, \ldots, F_n$ for a set of software components $1, \ldots, n$) [13]. A validation test of M with respect to F is shown at point 3 in Fig. 1. We denote M' as a validated metric. Since M is validated with respect to F, it is necessarily the case that F is valid. Therefore we say that F is valid by definition as a result of wide acceptance or historical usage (e.g., error count).

Since F is a direct measure of quality, it is preferred over M whenever it is possible to measure F sufficiently early in the life cycle to permit quality to be assessed, controlled, and predicted (see below). However, since this is usually not the case, the need for validation arises. We also note that since the cost of finding and correcting errors grows rapidly with the life cycle, it is advantageous to have approximate early (leading) indicators of software quality. (Analogously, one could posit that the Dow Jones stock price average (M) is an approximate leading indicator of the gross national product (F) in the American economy and conduct a validation test between the two.) Thus we can formulate the following policy with respect to software measurement: when it is feasible to measure and apply F, use it; otherwise, attempt to validate M with respect to F and, if successful, use M'.

D. Quality Functions

Quality functions are activities conducted by software organizations for the purpose of achieving project quality goals.

Both product and process goals are included. The quality functions that are pertinent to this metrics methodology are: assessment, control, and prediction.

1) Quality Assessment: Quality assessment is the evaluation of the relative quality of software components. "Relative quality" is the quality of a given component compared with the quality of other components in the set (e.g., if M' is cyclomatic complexity, the quality of component 1, with $M' = 3$, may be better than the quality of component 2, with $M' = 10$). Validated metrics are used to make a relative comparison of the quality of software components. The purpose of assessment is to provide software managers with a rational basis for assigning priorities for quality improvement and for allocating personnel and computer resources to quality assurance functions. For example, priorities and resources would be assigned on the basis of relative values (or ranks) of M' (i.e., the most resources would be assigned to the components with the highest (lowest) values (or ranks) of M'). M' is shown collected at point 4 in Fig. 1, and used for assessment at point 5.

2) Quality Control: Quality control is the evaluation of software components against predetermined critical values of metrics (i.e., the value of M' that is used to identify software which has unacceptable quality [13]) and the identification of components that fall outside quality limits. We denote M'_c as the critical value of M'. Validated metrics are used to identify components with unacceptable quality. The purpose of control is to allow software managers to identify software that has unacceptable quality sufficiently early in the development process to take corrective action. For example, $M'_c = 3$ would be used as a critical value of cyclomatic complexity to discriminate between components that contain errors and those that do not.

Control also involves the tracking of the quality of a component over its life cycle. For example, if M' is cyclomatic complexity, an increase from 3 to 10, as the result of a design change, would be used to indicate possible degradation in quality. M' is shown as collected at point 4 in Fig. 1, and used for control at point 5.

3) Quality Prediction: Quality prediction is a forecast of the value of F at time $T2$ based on the values of $M'_1, M'_2, \ldots, M'_n$ for components $1, 2, \ldots, n$ at time $T1$, where "time" could be computer execution time, labor time, or calendar time. Validated metrics (e.g., size and complexity) are used during the design phase to make predictions of test or operational phase factors (e.g., error count). The purpose of prediction is to provide software managers with a forecast of the quality of the operational software, and to flag components for detailed inspection whose predicted factor values are greater than (or less than) the target values (determined from requirements analysis). M' is shown as collected at point 4 in Fig. 1, and used for prediction at point 5.

E. Validity Criteria

Validity criteria provide the rationale for validating metrics—they are the specific quantitative relationships that are hypothesized to exist between factors and metrics. Validity criteria, in turn, are based on the principle of validity, which defines the general quantitative relationship between factors and metrics that must exist for the validity criteria to be applied. First, we provide definitions relating to the principle of validity. Then we define the principle of validity. Lastly, we define each validity criterion and provide an example of its application.

1) Definitions:

$$R[M]: \text{Relation } R \text{ on vector } \boldsymbol{M} \text{ for } V[P1, T1, M] \quad (1)$$

$$R[F]: \text{Relation } R \text{ on vector } \boldsymbol{F} \text{ for } V[P1, T2, F] \quad (2)$$

$$R[M']: \text{Relation } R \text{ on vector } \boldsymbol{M'} \text{ for } A[P2, T1, M'] \quad (3)$$

$$R[F']: \text{Relation } R \text{ on vector } \boldsymbol{F'} \text{ for } A[P2, T2, F'] \quad (4)$$

where R could be, for example, an order relation such as: Magnitude $[M_1 < M_2 \ldots < M_n]$ and Magnitude $[F_1 < F_2 \ldots < F_n]$ involving n values (data points) for M and F.

2) Principle of Validity:

$$\text{IF } R[M] \Leftrightarrow R[F]$$

is validated statistically with confidence level α and, for certain validity criteria, with threshold value β_i,

$$\text{THEN } \{R[M] \Leftrightarrow R[F]\} \Rightarrow \{R[M'] \Rightarrow R[F']\}? \quad (5)$$

In other words, does the mapping $M \Leftrightarrow F$, validated on Project 1, imply a mapping $M' \Rightarrow F'$ on Project 2? We assume (5) to be true at point 5 in Fig. 1. Once F' is collected at point 6, we revalidate (or invalidate) (5) by repeating the validation test using aggregated M and M', validated with respect to aggregated F and F' at point 7.

We note that a metric may be valid with respect to certain validity criteria, and invalid with respect to other criteria. Each validity criterion supports one or more of the quality functions assessment, control, and prediction, which were described above. The validity criteria—association, consistency, discriminative power, tracking, predictability, and repeatability—are applied at point 3 of Fig. 1. The particular criteria that are used depend on the quality functions (one or more) that are to be supported.

The validation procedure requires that threshold values β_i be selected for certain validity criteria. The criterion used for selecting these values is reasonableness (i.e., judgment must be exercised in selecting values to strike a balance between the one extreme of causing an M, which has a high degree of association with F, to fail validation, and the other extreme of allowing an M of questionable validity to pass validation).

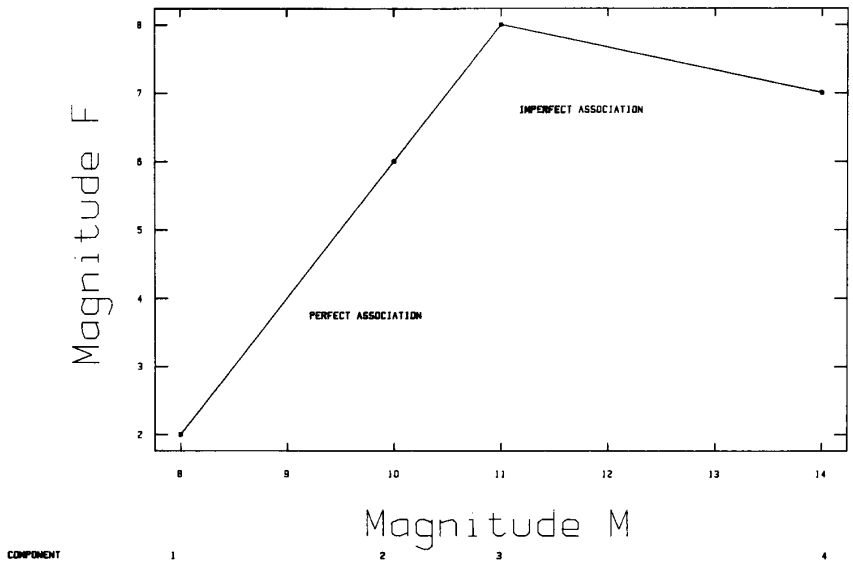

Fig. 2. Association validity criterion.

A short simple numerical example follows the definition of each validity criterion for the purpose of illustrating the basic concepts of the validity criteria. For illustrative purposes, F is error count and M is cyclomatic complexity, or complexity for short, in the examples. Also, to keep the examples simple we use small sample sizes; these sample sizes would not be acceptable in practice. As noted previously, given $\{F\}$ and $\{M\}$, it is possible to have an M_j in $\{M\}$ predict multiple F's in $\{F\}$, or to have an F_i in $\{F\}$ predicted by multiple M's in $\{M\}$. However, in order to simplify the examples, only the one-to-one case will be illustrated.

3) Association: The variation in F explained by the variation in M, which is given by R^2 (coefficient of determination), where R is the linear correlation coefficient, must exceed a specified threshold, or

$$R^2 > \beta_a, \text{ with specified } \alpha. \tag{6}$$

This criterion assesses whether there is a sufficient linear association between F and M to warrant using M as an indirect measure of F. This criterion supports the quality assessment function as follows.

If the elements of vector M, corresponding to components $1, 2, \ldots, n$, are ordered by magnitude, as illustrated in Table I, can we infer a linear ordering of F with respect to M for the purpose of assessing differences in component quality? In other words, does the following hold?

$$\text{Magnitude}[M_1 < M_2 \ldots < M_i \ldots < M_n] \Leftrightarrow$$
$$\text{Magnitude}[F_1 < F_2 \ldots < F_i \ldots < F_n] \tag{7}$$

and $(M_{i+1} - M_i) \propto (F_{i+1} - F_i)$ for $i = 1, 2, \ldots, n-1$.

The data of Table I are plotted in Fig. 2 to contrast perfect with imperfect association.

Since there is seldom perfect linear magnitude ordering between F and M (i.e., $R = 1.0$), we use (6) to measure the degree to which (7) holds. For example, if $R = 0.9$ and $\alpha = 0.05$, then 81% of the variation in F (error count) is explained by the variation in M (complexity), with an acceptable confidence level. If this relationship is demonstrated over a representative sample of components, and if β_a has been established as 0.7, we could conclude that M is associated with F and can be used to compare magnitudes of complexity obtained from different components to assess the degree to which they differ in quality (e.g., the difference in complexity magnitude between component 2 and component 1 (10–8) is proportional to their differences in quality in Table I).

TABLE I
VALIDATION PROJECT

Component	*M* (magnitude)	*M* (rank)	*F* (magnitude)	*F* (rank)
1	8	1	2	1
2	10	2	6	2
3	11	3	8	4
4	14	4	7	3

The resultant M' would be used to assess differences in the quality of components on the application project.

4) Consistency: The rank correlation coefficient r between F and M must exceed a specified threshold, or

$$r > \beta_c, \text{ with specified } \alpha. \tag{8}$$

This criterion assesses whether there is sufficient consistency between the ranks of F and the ranks of M to warrant using M as an indirect measure of F [9]. This criterion supports the quality assessment function as follows.

If the elements of vector M, corresponding to components $1, 2, \ldots, n$, are ordered by rank as illustrated in Table I, can we infer an ordering of F with respect to M for the purpose

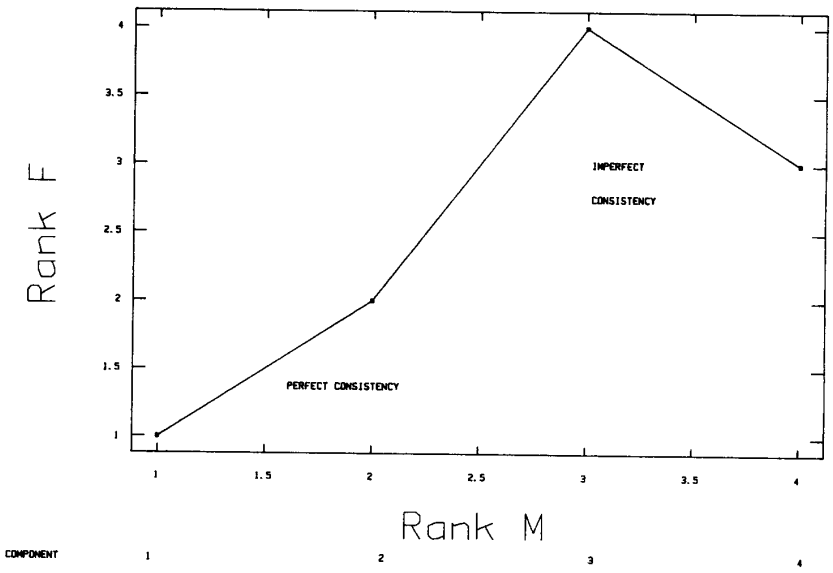

Fig. 3. Consistency validity criterion.

of assessing the rank order of component quality? In other words, does the following hold?

$$\text{Rank}[M_1 < M_2 \ldots < M_i \ldots < M_n] \Leftrightarrow \text{Rank}[F_1 < F_2 \ldots < F_i \ldots < F_n]. \quad (9)$$

The data of Table I are plotted in Fig. 3 to contrast perfect with imperfect consistency for the same set of components.

Since there is seldom perfect rank ordering between F and M (i.e., $r = 1.0$), we use (8) to measure the degree to which (9) holds. For example, if $r = 0.8$ and $\alpha = 0.05$, there is an 80% ranking between F and M, with an acceptable confidence level. If this relationship is demonstrated over a representative sample of components, and if β_c has been established as 0.7, we could conclude that M is consistent with F and can be used to compare ranks of complexity obtained from different components to assess the degree to which they differ in relative quality (e.g., component 2 quality is lower (higher complexity) than component 1 quality in Table I).

The resultant M' would be used to assess relative quality of components on the application project.

5) Discriminative Power: The critical value of a metric M_c must be able to discriminate, for a specified F_c, between elements (components $1, 2, \ldots, i, \ldots n$) of vector F [17] in the following way:

$$M_i > M_c \Leftrightarrow F_i > F_c \text{ and } M_i \leq M_c \Leftrightarrow F_i \leq F_c \quad (10)$$

for $i = 1, 2, \ldots, n$ with specified α.

This criterion assesses whether M_c has sufficient discriminative power to warrant using it as an indirect measure of F_c. This criterion supports the quality control function as follows.

Would M_c, as illustrated in Table II, partition F for a specified F_c as defined in (10)? For example, the data from Table I is used in Table II, with $M_c = 10$ and $F_c = 2$. We see that discriminative power is not perfect in Table II (i.e., $O_{21} \neq 0$). If it is desired to flag components with more than two errors ($F > F_c$) for detailed inspection and if $M_c' = 10$ (complexity) is validated, it would be used on the application project to control quality (i.e., discriminate between acceptable and unacceptable components), as shown in Fig. 4. One purpose of Fig. 4 is to identify trends in quality (e.g., a persistent case of components being in the unacceptable zone).

TABLE II
VALIDATION PROJECT

$M_c = 10$ $F_c = 2$	$M \leq M_c$	$M > M_c$
$F \leq F_c$	$O_{11} = 1$	$O_{12} = 0$
$F > F_c$	$O_{21} = 1$	$O_{22} = 2$

O_{ij} = count of observations in cell i, j.
O_{11}, O_{22}: correct classifications.
O_{12}, O_{21}: incorrect classifications.

Since there is seldom a perfect discriminator M_c for F_c (i.e., $O_{12} = O_{21} = 0$ in Table II), we use an appropriate statistical method (e.g., chi-square contingency table [7], [8], [12]) and representative sample of components to measure the degree to which (10) holds.

6) Tracking: M must change in unison with F for a given component i at times $T_1, T_2, \ldots T_j, \ldots, T_m$ as follows:

$$\begin{aligned} M_i(T_{j+1}) > M_i(T_j) &\Leftrightarrow F_i(T_{j+1}) > F_i(T_j) \\ M_i(T_{j+1}) = M_i(T_j) &\Leftrightarrow F_i(T_{j+1}) = F_i(T_j) \\ M_i(T_{j+1}) < M_i(T_j) &\Leftrightarrow F_i(T_{j+1}) < F_i(T_j) \end{aligned} \quad (11)$$

with specified α.

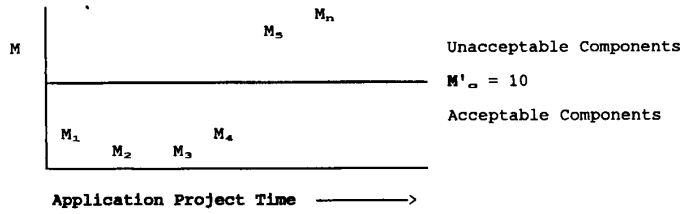

Fig. 4. Application of metrics to quality control (discriminative power) for components $1, 2, \ldots, n$.

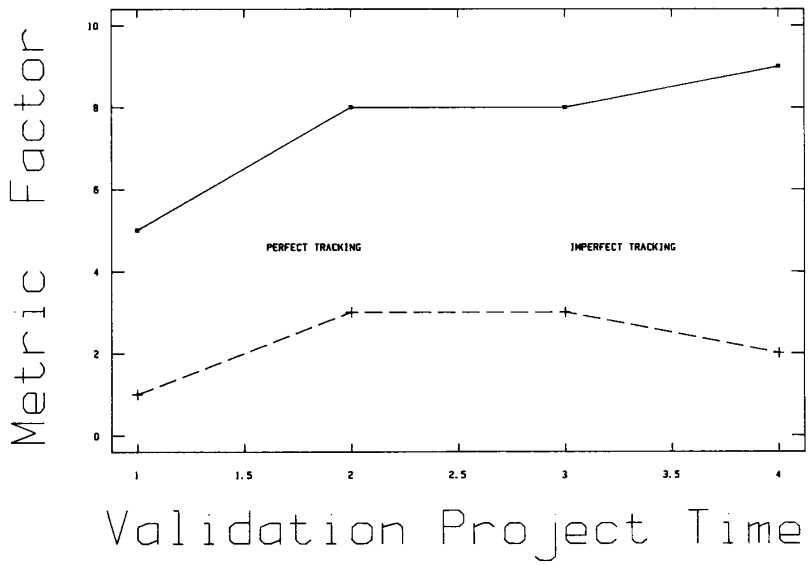

Fig. 5. Tracking validity criterion (component i).

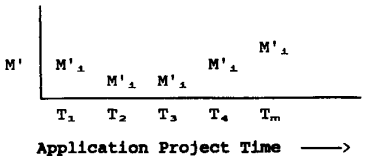

Fig. 6. Application of metrics to quality control (tracking) for component i at times $1, 2, \ldots, m$.

This criterion is illustrated graphically in Fig. 5 to contrast perfect with imperfect tracking, where factor and metric values are plotted against project time.

This criterion assesses whether M is capable of tracking changes in F (e.g., as a result of design changes) to a sufficient degree to warrant using M as an indirect measure of F. This criterion supports the quality control function as follows.

Would changes in M track changes in F as defined in (11)? If M is validated, then a vector $M_i'(T_j)$ consisting of the values $M_i'(T_1), M_i'(T_2), \ldots, M_i'(T_j), \ldots, M_i'(T_m)$ of component i, measured at times $T_1, T_2, \ldots, T_j, \ldots, T_m$ would be used to track quality on the application project. For example, if complexity M_i' is valid for tracking error count F, M_i' would be used as shown in Fig. 6, where quality increases from T_1 to T_2, stays the same from T_2 to T_3, and decreases thereafter.

Since there is seldom perfect tracking of F by M, we use an appropriate statistical method (e.g., binary sequences test [8]) and representative sample for component i to measure the degree to which (11) holds.

7) Predictability: A function of M, $f(M)$, where M is measured at time $T1$, must predict F, measured at time $T2$, with an accuracy β_p or

$$\left|\frac{Fa_{T2} - Fp_{T2}}{Fa_{T2}}\right| < \beta_p \tag{12}$$

where Fa_{T2} is the actual value, and Fp_{T2} is the predicted value.

This criterion is illustrated graphically in Fig. 7 to contrast perfect with imperfect prediction, where $f(M)$, formulated at $T1$, will either turn out to be equal to Fa at $T2$ (perfect Predictability), or be equal to Fp^+ or Fp^- (imperfect Predictability).

This criterion assesses whether $f(M)$ can predict F with required accuracy. This criterion supports the quality prediction function as follows.

If (12) holds, would the following hold?

$$Fp_{T2} = f(M_{T1}) \Rightarrow Fp'_{T2} = f(M'_{T1}) \tag{13}$$

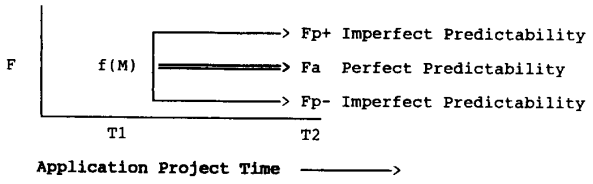

Fig. 7. Application of metrics to quality prediction (predictability) for a component.

where vector $Fp_{T2} = [F_1, F_2, \ldots, F_n]_{T2}$ and vector $M_{T1} = [M_1, M_2, \ldots, M_n]_{T1}$ for components $1, 2, \ldots, n$, and Fp'_{T2}, and M'_{T1} are similarly defined. In other words, do we have:

$$\left| \frac{Fa'_{T2} - Fp'_{T2}}{Fa'_{T2}} \right| < \beta_p. \tag{14}$$

For example, if a function f relating error count with complexity can be identified (e.g., regression analysis) that is a good predictor of F (i.e., satisfies (12)), then we would use the same f as the predictor of F' to predict error count from complexity on the application project.

Since there is seldom a perfect f (i.e., $Fp_{T2} = Fa_{T2}$), we use (12) to measure the degree to which f predicts F.

8) Repeatability: The success rate of validating M for a given validity criterion i must satisfy:

$$N_{is}/N_i > \beta_{is} \tag{15}$$

where N_{is} is the number of validations of M for criterion i, and N_i is the total number of trials for criterion i.

This criterion assesses whether M can be validated on a sufficient percentage of trials to have confidence that it would be a dependable indicator of quality in the long run. We use "trials," because validation could be performed with respect to projects, applications, components, or some other appropriate entity.

F. Metrics Validation Process

Given that there must be a validation project V and an application project A, as shown in Fig. 1, this requirement gives rise to what we call the "fundamental problem in metrics validation." This problem arises because there could be significant time lags, product and process differences, and differences in goals and environments [5] between the following phases of the validation process (see Fig. 1):

1) $V[P1, T1, M]$ and $V[P1, T2, F]$
2) $V[P1, T2, F]$ and $A[P2, T1, M']$
3) $A[P2, T1, M']$ and $A[P2, T2, F']$.

An important characteristic of the methodology is expressed by the following:

$$\text{IF } V[P1, T1, M] \Leftrightarrow V[P1, T2, F]$$
$$\text{THEN } A[P2, T1, M'] \Rightarrow A[P2, T2, F']. \tag{16}$$

From (16), it follows that at point 3 in Fig. 1, M is validated in V. Whether M' will actually be valid in A will not be known until point 7. Thus it is worthwhile to discuss some of the practical difficulties of adhering to (16) and possible remedies.

With respect to phase 1), the product or process may have changed so much between $T1$ and $T2$ that M, collected at $T1$, may no longer be representative of F. If this is the case, M should be collected again at $T2$ to validate against F. The advantage of collecting M at $T1$ is that it may be easier and less expensive than at $T2$, because M can be collected as a by-product of compilation and design and code inspections.

The same considerations apply with respect to phase 3), except that now the concern is with whether M' collected at $T1$ should be used for revalidation at $T2$. However, note that it is mandatory that M' be collected at $T1$ to have an early indication of possible quality problems (that is a key concept of our methodology!).

With respect to phase 2), we can achieve a degree of stability in the validation process if the following procedure is employed:

a) Select V and A to be as similar as possible with respect to application and development environments.

With respect to phases 1)–3) considered jointly, we can achieve a degree of stability in the validation process if procedure a) is employed, plus the following two additional procedures:

b) Select the same life cycle phase for $T1$ in V and A.
c) Select the same life cycle phase for $T2$ in V and A.

We recognize that it may be infeasible to implement these procedures. If this is the case, it means that there is a higher risk that M validated at point 3 in Fig. 1 will not remain valid at point 5.

III. Nonparametric Statistical Methods for Metrics Validation

Nonparametric statistical methods are used to support metrics validation, because these methods have important advantages over parametric methods. Indeed, it would be infeasible to validate metrics in many situations without their use. This is the case, because the assumptions that must be satisfied to employ nonparametric methods are less demanding than those that apply to parametric methods. This might lead to the conclusion that nonparametric methods are less rigorous than parametric methods. Despite this possible perception, nonparametric methods allow us to develop very useful order relations concerning the relative quality of components. The validity criteria which use nonparametric methods are shown in Table III. The advantages of nonparametric methods over parametric methods, which are important for metrics validation, are the following:

TABLE III
VALIDITY CRITERIA PROPERTIES

Criterion	Scale	Method	Measurement Property
Association	Interval	Parametric	Difference
Consistency	Ordinal	Nonparametric	Higher/Lower
Discriminative Power	Nominal	Nonparametric	High/Low
Tracking	Nominal	Nonparametric	Increment
Predictability	Interval, Ratio	Parametric	% Accuracy
Repeatability	Ratio	Parametric	% Success

TABLE IV
EXAMPLE DATA

Project Application	Procedures (with errors)	Statements	Errors
1. String Processing	11 (5)	136	10
2. Directed Graph Analysis	31 (12)	430	27
3. Directed Graph Analysis	1 (1)	13	1
4. Data Base Management	69 (13)	1021	26
	112 (31)	1600	64

Number of procedures: 112 total, 31 with errors, 81 with no errors.
Number of source statements: 2007 total, 1600 included in metrics analysis.
Language : Pascal on all projects.
Programmer: Single programmer. Same programmer on all projects.

- Given the noisiness of metrics data, the fact that the assumptions are less restrictive is a big advantage.
- No assumption is necessary about distribution (e.g., data does not have to be normally distributed).
- We can use the nominal scale (i.e., component A is high quality, component B is low quality) and location statistics like the median [11]. The *Discriminative Power* validity criterion is based on this measurement property. Similarly, we can use the nominal scale to indicate whether an incremental change in a metric tracks (yes/no) an incremental change in a factor. The *Tracking* validity criterion is based on this measurement property.
- We can use the ordinal scale (i.e., component A is higher quality than component B) and order statistics such as ranks. The *Consistency* validity criterion is based on this measurement property. For example, ranks of random variables [3] can be used rather than the values themselves, thus relaxing the assumptions about data relationships (e.g., linearity), while providing a measure of quality (e.g., ranking of components) that is useful to the software manager. In other words, the fact that the data is not as "well-behaved" as we might believe it should be does not necessarily mean that it is less useful. In fact, when we consider that many useful applications of metrics can be derived from the ability to classify components as being "higher quality" or "lower quality," we realize that the information provided by nonparametric analysis is supportive of this approach.

TABLE V
CONTINGENCY TABLE

	Complexity ≤ 3	Complexity > 3	
No Errors	75	6	81
Errors	10	21	31
	85	27	112

TABLE VI
PROJECTS 1, 2, 3, AND 4

C_c	χ^2	α
1	22.32	2.30E-6
2	32.14	1.44E-8
3	**41.60**	**1.26E-10**
4	26.80	2.26E-7

112 Procedures (81 with no errors, 31 with errors)

Despite the advantages of nonparametric methods, certain validity criteria lend themselves to the use of parametric methods. These are shown in Table III. "Association," which measures the difference in component quality, uses the interval scale. "Predictability" uses the interval scale to predict a factor value and the ratio scale for measuring prediction accuracy. Lastly, "Repeatability" uses the ratio scale for measuring metric validation success.

Appendix A summarizes the quality function, validity criterion, purpose of valid metric, and statistical method.

IV. EXAMPLE OF VALIDATING METRICS

The following example is provided to illustrate the validation of M with F and the identification of an M_c which would be used in the quality control function. Also, we show how to conduct a cost-sensitivity analysis on M_c in order to identify its optimal value (i.e., the minimum cost M_c across a range of assumptions about the cost of using M_c).

The data used in the example validation tests were collected from actual software projects. The "Discriminative Power" validity test is illustrated.

A. Purpose of Metrics Validation

The purpose of this validation is to determine whether cyclomatic number (complexity (C)) and size (number of source statements (S)) metrics, either singly or in combination, could be used to control the factor reliability as represented by the factor error count (E). A summary of the data is shown

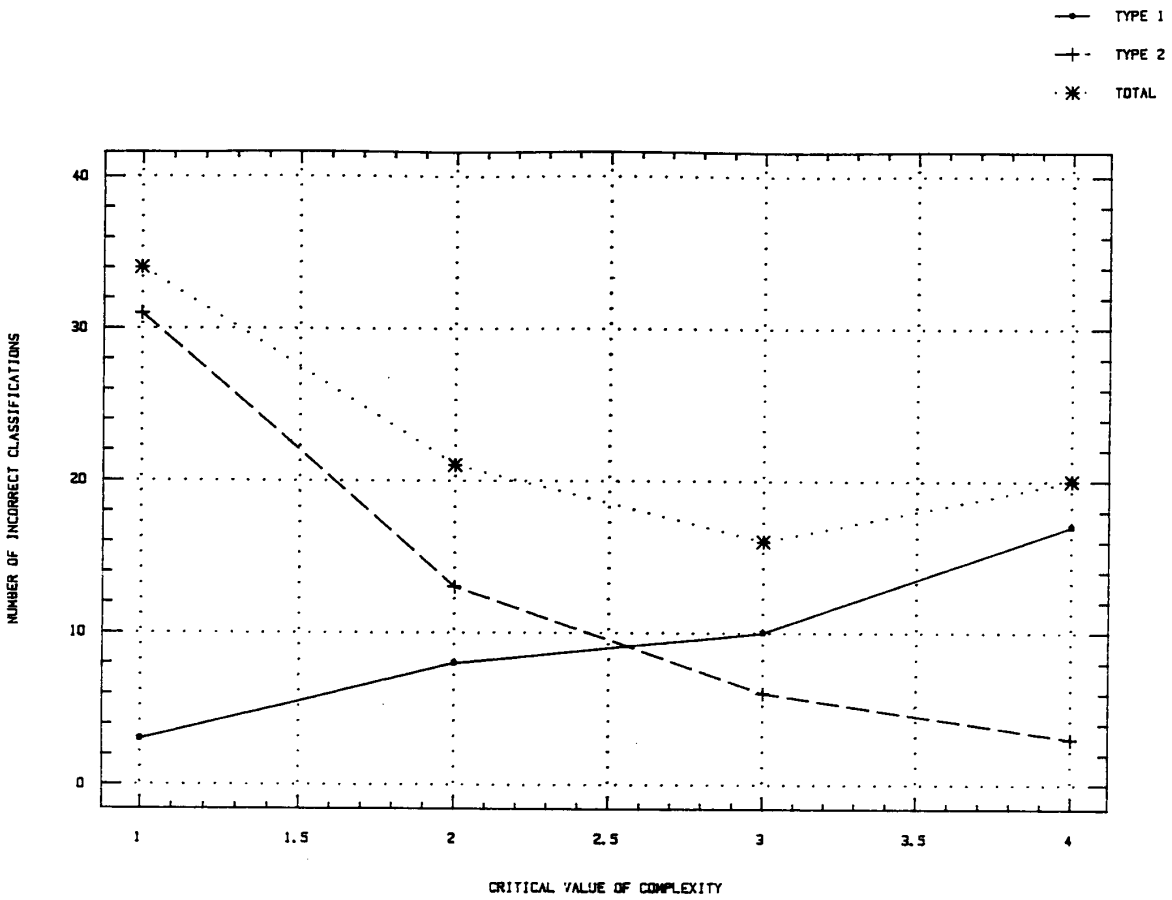

Fig. 8. Incorrect classification (complexity).

in Table IV, and the detailed data listing can be found in Appendix B.

Using the conventions of Fig. 1, the following is the notation applicable to this example:

Metric: C, S collected at point 1, Fig. 1.
Factor: E, collected at point 2, Fig. 1.
Critical Value of Metric: C_c, S_c validated at point 3, Fig. 1.
V[Projects 1, 2, 3, 4: Design; C, S]
V[Projects 1, 2, 3, 4: Test; E]

B. Discriminative Power Validity Test

We divide the data into four categories, as shown in Table V, according to a critical value of C, C_c, so that a chi-square test can be performed to determine whether C_c can discriminate between procedures with errors and those with no errors [4].

From the high value of chi-square (41.60) (see Table VI) and the very small significance level (1.26E-10) in the samples, we infer that $C_c = 3$ could discriminate between procedures with errors (low-quality software) and those without errors (high-quality software).

Table V shows how good a job $C_c = 3$ does to discriminate between procedures with errors and procedures with no errors: 75 of 81 with no errors, and 21 of 31 with errors are correctly classified.

C. Sensitivity Analysis of Critical Value of Complexity

In order to see how good a discriminator C_c is for this example, we observe the number of misclassifications that result for various values of C_c : (i) Type 1 ("error procedures," classified as "no error procedures"), and (ii) Type 2 ("no error procedures," classified as "error procedures"). This is shown in Fig. 8. As C_c increases Type 1 misclassifications increase, because an increasing number of high complexity procedures, many of which have errors, are classified as having "no errors." Conversely, as C_c decreases Type 2 misclassifications increase, because an increasing number of low complexity procedures, many of which have no errors, are classified as having "errors." The total of the two curves represents the "misclassification function." It has a minimum at $C_c = 3$, which is the value given by the chi-square test (see Table VI). The chi-square test will not always produce the optimal C_c,

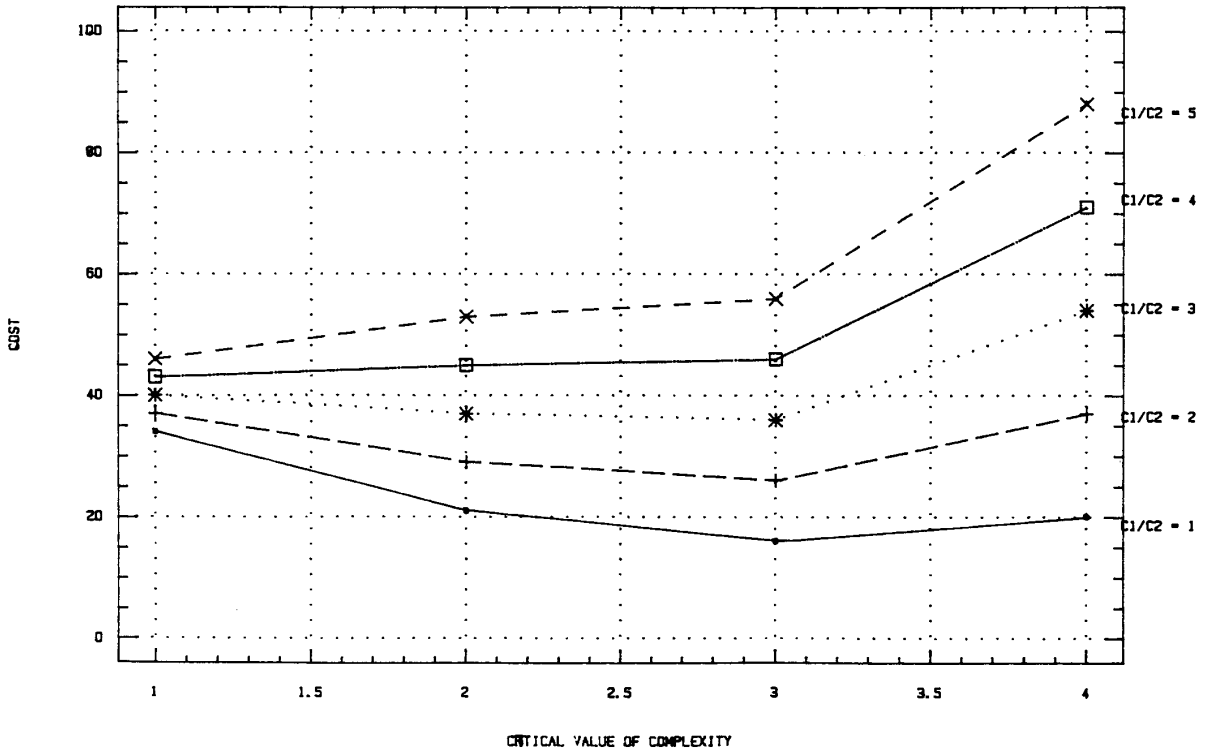

Fig. 9. Cost of incorrect classification (complexity).

but the value should be close to optimal.

The foregoing analysis assumes that the costs of Type 1 and Type 2 misclassifications are equal. This is usually not the case, since the consequences of not finding an error (i.e., concluding that there is no error when, in fact, there is an error) would be higher than the other case (i.e., concluding that there is an error when, in fact, there is no error). In order to account for this situation, the number of Type 1 misclassifications for given values of C_c is multiplied by $C1/C2$ ($C1/C2 = 1, 2, 3, 4, 5$), which is the ratio of the cost of Type 1 misclassification to the cost of Type 2 misclassification. These values are added to the number of Type 2 misclassification to produce the family of five "cost" curves shown in Fig. 9. Naturally, with the higher cost of Type 1 misclassifications taking effect, the optimal C_c (i.e., minimum cost) decreases. However, even at $C1/C2 = 5, Cc = 3$ is a reasonable choice.

A "Contingency Table" was also developed for S, leading to $S_c = 13$. The same type of sensitivity analysis was performed on S_c. It was found that the optimal is $S_c = 15$, as opposed to $S_c = 13$, as given by the chi-square analysis.

We conclude that C and S are valid with respect to the "Discriminative Power" criterion, and either could be used to distinguish between acceptable ($C \leq 3, S \leq 13$) and unacceptable quality ($C > 3, S > 13$) for this and similar applications when this data can be collected. However, only one is needed (i.e., C is highly correlated with S). It should be noted that it is less expensive to collect S than C.

V. Summary and Future Research

We described and illustrated a comprehensive metrics validation methodology that has six validity criteria, which support the quality functions of assessment, control, and prediction. Six criteria were defined and illustrated: association, consistency, discriminative power, tracking, predictability, and repeatability. These criteria are important because they provide a rationale for validating metrics; in practice, this rationale is frequently lacking in the selection and application of metrics. With validated metrics we have a basis for making decisions and taking actions to improve the quality of software. We showed that quality factors, metrics, and functions can be integrated with our metrics validation process. We developed a framework which pulls together the concepts and definitions of quality factor, quality metric, validated metric, quality function, validity criteria, and the metrics validation process. We showed that nonparametric statistical methods play an important role in evaluating whether metrics satisfy the validity criteria. An example of the application of the methodology was presented for the discriminative power validation criterion. The discriminative power criterion allows the metrics user to control the production of highly reliable software by providing thresholds of acceptable quality.

Future research is needed to extend and improve the methodology by finding an answer to the following question: to what extent are metrics that have been validated on one project, using our criteria, valid measures of quality on future projects—both similar and different projects?

APPENDIX A

Appendix A is given in Table VII.

TABLE VII
APPENDIX A

Quality Function	Validity Criterion	Purpose of Valid Metric	Statistical Method
Quality Assessment	Association	Assess differences in quality	1. Coeff. of Determination $R^2 > \beta_a$. 2. H0: Population Correlation Coeff. = 0. 3. H0: Population Correlation Coefficient $> \sqrt{\beta_a}$. 4. Linear Partial Correlation Coeff. (Metric Normalization. Accounting for Size). 5. Population Correlation Coefficient Confidence Interval. 6. Factor Analysis (Tests of Independence).
Quality Assessment	Consistency	Assess relative quality	1. Rank Correlation Coefficient $r > \beta_c$.
Quality Control	Discriminative Power	Control Quality (discriminate between high and low)	1. Mann-Whitney Comparison of Average Ranks of Two Groups of components. 2. Chi-square Contingency Table for Finding Critical Value of Metric. 3. Short-Cut Technique for Finding Critical Value of Metric: Maximize $O_{11}O_{22}$. 4. Sensitivity Analysis of Critical Value of Metric. 5. Krusal-Wallis Test of Average Metric Rank Per Given Value of Quality Factor. 6. Discriminant Analysis (Use of a Single Metric's Mean as Discriminator).
Quality Control	Tracking	Control quality (track changes)	1. Binary Sequences Test and Wald-Wolfowitz Runs Test.
Quality Prediction	Predictability	Predict quality	1. Scatter Plot to Investigate Linearity. 2. Linear Regression. a. Test Assumptions b. Examine Residuals 3. Find Confidence and Prediction Intervals. 4. Test for Predictability < Threshold (β_p) and Repeatability > Threshold (β_{is}). 5. Non-linear Regression. 6. Multiple Linear Regression. a. Test Assumptions b. Examine Residuals c. Test for Predictability < Threshold (β_p) and Repeatability > Threshold (β_{is}).
All Quality Functions	Repeatability	Ensure metric validated with specified success rate	Ratio of Validations to Total Trials > Threshold (β_{is})

APPENDIX B

Appendix B is given in Table VIII.

TABLE VIII
APPENDIX B—PROCEDURES WITH NO ERRORS

C	S	E	Project	C	S	E	Project
2	6	0	1	1	3	0	4
1	8	0	1	1	3	0	4
1	11	0	1	1	3	0	4
1	4	0	1	1	5	0	4
3	18	0	1	1	5	0	4
3	15	0	1	1	6	0	4
1	3	0	2	1	9	0	4
1	3	0	2	1	6	0	4
1	3	0	2	1	8	0	4
1	3	0	2	1	9	0	4
1	3	0	2	1	9	0	4
1	3	0	2	2	4	0	4
1	3	0	2	2	7	0	4
1	3	0	2	2	9	0	4
1	5	0	2	4	56	0	4
1	5	0	2	1	24	0	4
1	5	0	2	2	13	0	4
1	13	0	2	2	13	0	4
1	3	0	2	2	10	0	4
1	3	0	2	2	9	0	4
1	3	0	2	2	12	0	4
1	3	0	2	5	21	0	4
1	3	0	2	5	49	0	4
1	3	0	2	3	19	0	4
1	3	0	2	4	20	0	4
1	2	0	4	2	6	0	4
1	2	0	4	2	12	0	4
1	7	0	4	2	9	0	4
1	5	0	4	2	10	0	4
1	7	0	4	1	21	0	4
1	5	0	4	4	21	0	4
1	5	0	4	3	11	0	4
1	5	0	4	2	13	0	4
1	5	0	4	3	14	0	4
1	4	0	4	7	19	0	4
1	3	0	4	2	15	0	4
1	3	0	4	2	10	0	4
1	3	0	4	2	17	0	4
1	3	0	4	3	19	0	4
1	3	0	4	3	15	0	4
				2	15	0	4
Procedures with Errors							
2	14	1	1	4	26	1	2
6	26	5	1	16	94	8	2
5	7	2	1	2	13	1	3
5	21	1	1	6	83	1	4
2	6	1	1	5	28	1	4
1	3	1	2	8	37	5	4
1	11	1	2	3	13	2	4
1	8	1	2	3	16	1	4
2	15	3	2	7	34	1	4
8	45	3	2	5	24	1	4
4	18	1	2	4	18	3	4
6	54	3	2	5	35	2	4
2	34	2	2	13	49	5	4
4	19	1	2	4	19	1	4
5	30	2	2	4	27	1	4
				4	17	2	4

(Continued) TABLE VIII
APPENDIX B—PROCEDURES WITH NO ERRORS

C : Complexity
S : Number of source statements (excluding comments)
E : Error count

ACKNOWLEDGMENT

The author thanks the referees for their many useful comments and suggestions that have greatly improved this paper. He also thanks the members of the IEEE Standard for a Software Quality Metrics Methodology Working Group for many useful discussions and debates that helped inspire this work.

REFERENCES

[1] A. J. Albrecht and J. E. Gaffney, Jr., "Software function, source lines of code, and development error prediction: a software science validation," *IEEE Trans. Software Eng.*, vol. SE-9, pp. 639–648, Nov. 1983.
[2] A. L. Baker *et al.*, "A philosophy for software measurement," *J. Syst. Software*, vol. 12, no. 3, pp. 277–281, July 1990.
[3] V. R. Basili, R. W. Selby, Jr., and T.-Y. Phillips, "Metric analysis and data validation across Fortran projects," *IEEE Trans. Software Eng.*, vol. SE-9, pp. 652–663, Nov. 1983.
[4] V. R. Basili, and D. H. Hutchens, "An empirical study of a syntactic complexity family," *IEEE Trans. Software Eng.*, vol. SE-9, pp. 664–672, Nov. 1983.
[5] V. R. Basili and H. D. Rombach, "The TAME project: toward improvement-oriented software environments," *IEEE Trans. Software Eng.*, vol. 14, pp. 759–773, June 1988.
[6] M. E. Bush and N. E. Fenton, "Software measurement: a conceptual framework," *J. Syst. Software*, vol. 12, no. 3, pp. 223–231, July 1990.
[7] D. N. Card, G. T. Page, and F. E. McGarry, "Criteria for software modularization," in *Proc. 8th Int. Conf. on Software Eng.*, Aug. 1985, pp. 372–377.
[8] W. J. Conover, *Practical Nonparametric Statistics.* New York: Wiley, 1971.
[9] S. D. Conte, H. E. Dunsmore, and V. Y. Shen, *Software Engineering Metrics and Models.* Menlo Park, CA: Benjamin/Cummings, 1986.
[10] L. Felician and G. Zalateu, "Validating Halstead's theory for Pascal programs," *IEEE Trans. Software Eng.*, vol. 15, pp. 1630–1632, Dec. 1989.
[11] N. E. Fenton and A. Melton, "Deriving structurally based software metrics," *J. Syst. Software*, vol. 12, no. 3, pp. 177–187, July 1990.
[12] J. D. Gibbons, *Nonparametric Statistical Inference.* New York: McGraw-Hill, 1971.
[13] *IEEE Standard for a Software Quality Metrics Methodology* (draft), no. P-1061/D21, Apr. 1, 1990.
[14] *IEEE Standard Glossary of Software Engineering Terminology*, ANSI/IEEE Std. 729–1983.
[15] *IEEE Glossary of Software Engineering Terminology* (draft), no. P729/610.12/D8, Mar. 30, 1990.
[16] A. A. Porter and R. W. Selby, "Empirically guided software development using metric-based classification trees," *IEEE Software*, vol. 7, no. 2, pp. 46–54, Mar. 1990.
[17] E. J. Weyuker, "Evaluating software complexity measures," *IEEE Trans. Software Eng.*, vol. 14, pp. 1357–1365, Sept. 1988.

Norman F. Schneidewind (A'54–M'59–M'72–SM'77) received the B.S. degree in electrical engineering from the University of California, Berkeley, M.S. degrees in electrical engineering and computer science from San Jose State University, and the M.S. degree in operations research (engineering) and the Ph.D. degree (operations research) from the University of Southern California. He also holds the Certificate in Data Processing.

He is the developer of the Schneidewind Software Reliability Model, which is used by IBM-Houston

to report to NASA on the estimated time of occurrence of the next software failure for the Space Shuttle. This model is one of the models that is being recommended for use in the *American Institute of Aeronautics and Astronautics Software Reliability Estimation and Prediction Handbook*. He is a Professor of Information Sciences at the Naval Postgraduate School, Monterey, CA, and teaches and does research in the fields of software engineering and computer networks and is the Director of Computer Laboratories in his department.

Dr. Schneidewind is the Chairman of the IEEE Standard for the Software Quality Metrics Working Group, which has developed the first standard in this area. He is a member of the Department of Defense Software Engineering Institute Measurement Advisory Committee. He is Vice Chairman for standards of the Technical Committee on Software Engineering of the IEEE Computer Society, and is the Standards Editor for *IEEE Computer*. He is a member of the Eta Kappa Nu and Tau Beta Pi engineering honor societies, and the Sigma Xi research society.