1991

Setting Maintenance Quality Objectives and Prioritizing Maintenance Work by Using Quality Metrics

Schneidewind, Norman F.

IEEE

http://hdl.handle.net/10945/45151

Downloaded from NPS Archive: Calhoun
Setting Maintenance Quality Objectives and Prioritizing Maintenance Work by Using Quality Metrics

Norman F. Schneidewind

Code AS/Ss
Naval Postgraduate School
Monterey, CA 93943

Abstract

We show how metrics that are collected and validated during development can be used during maintenance to control quality and prioritize maintenance work. Our approach is to capitalize on knowledge acquired and experience gained with the software during development through measurement. The motivation for this research stems from the need to provide maintenance management with the following: 1) quantitative basis for establishing quality objectives during maintenance and 2) rationale for allocating resources - people, money, and equipment - to maintenance tasks. We base our approach on validating selected metrics against related quality factors during development and using the validated metrics during maintenance to: 1) establish initial quality objectives and quality control criteria and 2) prioritize software components (e.g., modules) and allocate resources to maintain them. We use the validity criteria discriminative power and tracking to illustrate the process.

1 INTRODUCTION

A comprehensive metrics validation methodology has been described in [Sch91, Sch90]. Six validity criteria were proposed and validation tests were illustrated with case studies from actual software: associativity, consistency, discriminative power, tracking, predictability and repeatability. Two of the criteria - discriminative power and tracking - can be used in maintenance to: 1) establish quality control objectives; and 2) prioritize software components (e.g., modules) and allocate resources to maintain them. These criteria are used because discriminative power measures the ability of a metric to discriminate between levels of quality (e.g., "high" and "low") during maintenance and tracking measures the ability of a metric to follow changes in quality that result from maintenance actions. In addition, these criteria are compatible with non-parametric statistical tests; these tests require minimal assumptions about the characteristics of the data (this is important in metrics analysis because the data are typically "noisy").

We show how metrics that are collected and validated during development can be used during maintenance to control quality and prioritize maintenance work. Validity criteria are defined mathematically in the "Validity Criteria" section. The example in the "Example of Validating Metrics" section illustrates both a case of passing a validation test (discriminative power) and failing a validation test (tracking).

The reasons for prioritizing maintenance tasks are to: 1) provide maintenance management with an objective and defensible procedure for allocating resources to perfective maintenance that is designed to reduce complexity and improve reliability and maintainability; and 2) provide a priority procedure for performing corrective maintenance. We recognize that in certain cases the business or mission objectives of the organization will take precedence over any priority procedure derived from metrics.

Measurement is continued during
maintenance and the initial quality objectives are subject to revision as more experience is gained with the software and as the metrics are revalidated. We recognize that software that is maintained may not resemble the software that is produced. However, our methodology protects against this possibility by requiring revalidation of metrics during maintenance. Thus, either the original hypothesis about the validity of the metrics will be reinforced or the metrics will be invalidated and abandoned.

2 RATIONALE FOR METRICS VALIDATION

To help ensure that metrics are used appropriately, only validated metrics (i.e., either quality factors (attributes of software that contributes to its quality [IEE90]) or metrics validated with respect to quality factors) should be used. Quality factors and metrics are "direct measures" and "indirect measures", respectively and the latter is used to "predict" or make an assessment about the former [Rom90]. Quality factors are valid by definition. Furthermore, the metrics which are used should be those which are associated with the quality requirements of the software project. In general, both product and process metrics are used to assess software quality, although the example in the "Example of Validating Metrics" section will be limited to product metrics.

3 QUALITY FUNCTIONS

In [Sch91, Sch90] metrics are applied in three major quality functions: quality assessment, quality control and quality prediction. If metrics are to aid in making decisions about software quality, the user of metrics must understand how this tool supports major quality functions in a software engineering organization. Metrics should not be validated unless the applications of metrics are clearly understood. Quality control is the function which is related to the validity criteria discriminative power and tracking. Therefore, we describe the need to validate metrics during development for application to the quality control function during maintenance (i.e., the relationship must be made between quality function and validity criteria).

3.1 QUALITY CONTROL

Discriminative Power

Metrics are used to monitor the condition of a component to determine whether the component appears to be out of tolerance. This is defined to be a component whose quality is below standard. This implies that critical values of metrics must be established prior to the monitoring activity for comparing against the measured values derived from the component. Measurements from a validated metric are compared with the critical values of the metrics. Components whose measurements are greater than (or less than) the critical values are flagged for detailed inspection. This concept is illustrated in Figure 1 for metric vector M for components 1, 2, ..., n. The role of metrics validation for this use of quality control is to identify a critical value of a metric, where the metric used in maintenance has been validated against a quality factor during development (i.e., conclude that a statistically significant relationship exists between the metric and a quality factor). Such a metric satisfies the discriminative power validity criterion.

<table>
<thead>
<tr>
<th>Mn</th>
<th>Unacceptable Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>Critical Value of Metric</td>
</tr>
<tr>
<td>M1</td>
<td>Acceptable Region</td>
</tr>
<tr>
<td>N2</td>
<td>M3</td>
</tr>
</tbody>
</table>

Maintenance Phase (Project Time →)

Figure 1. Application of Metrics to Quality Control (discriminative power)
Tracking

In addition to component quality lying within acceptable bounds, a desirable condition is for quality to improve over the life of the component (i.e., a component should exhibit quality growth). Thus, during all phases of the life of the component we wish to track quality in order to control quality. That is, we want to know whether the software is getting better, worse, or staying the same. This concept is illustrated in Figure 2 for metric vector $M$ for a given component $i$, measured at times $T_1, T_2, ..., T_n$. In this illustration, quality increases from $T_1$ to $T_2$, stays the same from $T_2$ to $T_3$, and decreases from $T_3$ to $T_n$, assuming high metric values are "bad". Here, the question for metrics validation is whether a metric can be identified whose changes over time will track changes in quality. In particular, if a metric has been validated as tracking a quality factor during development, it would serve for tracking quality during maintenance. Such a metric satisfies the tracking validity criterion.

<table>
<thead>
<tr>
<th>$M$</th>
<th>$M_i$</th>
<th>$M_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_1$</td>
<td>$T_2$</td>
<td>$T_3$</td>
</tr>
</tbody>
</table>

Maintenance Phase (Project Time $\longrightarrow$)

Figure 2. Application of Metrics to Quality Control (tracking)

4 FUNDAMENTAL PROBLEM IN METRICS VALIDATION AND APPLICATION

In [Sch91, Sch90] we explained the metrics validation "(V) - application (A) process" with the triple: [Project, Time, Measurement]. The validation activity V involves collecting the metric $M$ during phase $T_1$ on project $P_1$ and performing a validation test of $M$ against the quality factor $F$ collected during phase $T_2$ on the same project, where $T_2$ is later than $T_1$. The application activity A involves controlling the quality of project $P_2$ by collecting and applying the validated metric $M$ from V during phase $T_1$. Lastly, a retrospective analysis is performed, once quality factor $F$ is collected during phase $T_2$ on project $P_2$, to see how representative $M$ was of $F$; also, this part of the process involves a revalidation test of $M$ against $F$, using the aggregated data collected for $M$ and $F$ during $P_1$ and $P_2$.

Four triples comprise the process as follows:

- $V[P_1, T_1, M]$
- $V[P_1, T_2, F]$
- $A[P_2, T_1, M]$
- $A[P_2, T_2, F]$

where $P$, $T$, $M$ and $F$ indicate project, time, metric and quality factor, respectively.

Example:

- $V[1, Design, Complexity]$
- $V[1, Test, Error Count]$
- $A[2, Design, Complexity]$
- $A[2, Test, Error Count]$

Fortunately, in the case of maintenance, the above model can be significantly simplified because maintenance occurs on the same project and therefore validation can be performed on the same project in which the validated metrics are applied. Thus we modify the model and example as follows:

- $V[P, T_1, M]$
- $V[P, T_1, F]$
- $A[P, T_2, M]$
- $A[P, T_2, F]$

- $V[1, Debugging, Complexity]$
- $V[1, Debugging, Error Count]$
- $A[1, Maintenance, Complexity]$
- $A[1, Maintenance, Error Count]$

The steps in collecting, measuring, validating, applying, and revalidating...
are as follows:

1. Collect and measure \( V[P, T_1, M] \).
2. Collect and measure \( V[P, T_1, F] \).
3. Validate \( V[P, T_1, M] \) against \( V[P, T_1, F] \).
5. Apply \( A[P, T_2, M] \).
7. See whether \( A[P, T_2, M] \) is a good representation of \( A[P, T_2, F] \).

A consequence of the above are the following implications:

\[
\text{IF } V[P, T_1, M] \implies V[P, T_1, F] \quad \text{THEN } A[P, T_2, M] \implies A[P, T_2, F]
\]

The fundamental problem of validating and applying metrics to maintenance is the following: There must be a project phase in which metrics are validated (\( V \)) and a project phase in which these metrics are applied (\( A \)). The problem arises because: There could be significant time lags, product differences, and process differences between (\( V \)) and (\( A \)), thus signalling the need to exercise care in choosing (\( V \)) and (\( A \)) so that the application of validated metrics will be appropriate (valid!).

5 VALIDITY CRITERIA

To be considered valid, a metric must demonstrate a high degree of association with the quality factor it represents. A metric may be valid with respect to certain validity criteria and invalid with respect to other criteria.

**Discriminative Power:** A metric must be able to discriminate between high quality components (e.g., high MTTF) and low quality components (e.g., low MTTF). For example, the set of metric values associated with the former should be significantly higher (or lower) than those associated with the latter.

This criterion assesses whether a metric is capable of separating a set of high quality components from a set of low quality components. This capability allows one to establish critical values for metrics which can be used to identify components which may have unacceptable quality. This criterion supports the quality control function. The following non-parametric statistical methods can be used for this validation test: Mann-Whitney Test [Bas83, Con71, Con86, Gib71], chi-square test for differences in probabilities (contingency tables) [Con71, Gib71] and the Krusal-Wallis Test [Bas83, Con71, Con86, Gib71].

**Tracking:** If a metric \( M \) is directly related to a quality factor \( F \), for a given component, then a change in a quality factor value from \( F_t \), to \( F_{t+1} \), at times \( T_1 \) and \( T_2 \), must be accompanied by a change in metric value from \( M_t \), to \( M_{t+1} \), which is the same direction (e.g., if \( F \) increases, \( M \) increases). If \( M \) is inversely related to \( F \), then a change in \( F \) must be accompanied by a change in \( M \) in the opposite direction (e.g., if \( F \) increases, \( M \) decreases).

This criterion assesses whether a metric is capable of tracking changes in quality over the life of a component. This criterion supports the quality control function. The following non-parametric statistical methods can be used for this validation test: Binary Sequence Test [Sta87] and the Wald-Wolfowitz Runs Test (test for randomness) [Con71, Gib71]

**Validate and Apply Metrics in Similar Environments**

There have been great disparities in results reported in the literature.
concerning "relationships" between
metrics and the quantities they purport
to measure. For example, correlation
coefficients of number of errors with
Halstead Effort and McCabe Complexity
differ by a factor of almost two
[IEE90]. Differences have also been
reported with respect to specification
refinement levels [Hen90]. These
disparities point up the need to apply
metrics under conditions that are
similar to those used to validate the
metrics.

Revalidate Metrics

Metrics validation is a continuous
process. It is important to revalidate a
metric each time it is used. As the
software engineering process changes,
the validity of metrics changes. A
validated metric may not necessarily be
valid in other environments or
applications. A metric that has been
invalidated may be valid in other
environments or applications. Also it
can be the case that a metric will be
valid across projects but that its
critical value could be different for
each project.

6 EXAMPLE OF VALIDATING METRICS

The data used in the example
validation tests were collected from
actual software projects. The
discriminative power and tracking
validation tests are illustrated.

Purpose of Metrics Validation

In this example we illustrate the
validation process as applied to
maintenance. For this application we
want to determine whether cyclomatic
number (complexity (C)) and size (number
of source statements (S)) metrics,
either singly or in combination, could
be used during maintenance to control
the quality factor reliability, as
represented by the quality factor error
count (E). It is not our purpose to be a
proponent or an opponent of given
metrics. The validation results could be
different in other applications and
environments. However, there is evidence
to suggest that relatively high
complexity (e.g., cyclomatic number,
size) is frequently associated with
relatively high error counts [Lew89],
with the implication that these
components may be difficult to maintain.

6.1 VALIDATION PROCEDURE

Identify the Quality Factor Sample

During the debugging phase draw a
random sample of procedures (i.e.,
components), which is summarized in
Table 1, from the metrics data base, for
the quality factor reliability, which is
represented by the quality factor error
count (Errors).

Identify the Metrics Sample

During the debugging phase, using the
same procedures (i.e., components) in
Table 1, identify the metrics samples
for cyclomatic number (complexity) and
size (statements).

<table>
<thead>
<tr>
<th>Project</th>
<th>Application</th>
<th>Total Procedures</th>
<th>Procedures With Errors</th>
<th>Statements</th>
<th>Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>A</td>
<td>TP</td>
<td>WE</td>
<td>S</td>
<td>E</td>
</tr>
<tr>
<td>1 String Processing</td>
<td>11</td>
<td>5</td>
<td>136</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>2 Directed Graph Analysis</td>
<td>31</td>
<td>12</td>
<td>430</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>3 Directed Graph Analysis</td>
<td>1</td>
<td>1</td>
<td>13</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4 Data Base Management</td>
<td>69</td>
<td>13</td>
<td>1021</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>112</td>
<td>31</td>
<td>1600</td>
<td>64</td>
</tr>
</tbody>
</table>

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.
Perform Goodness of Fit Tests

The best fits obtained for the data are the following distributions:

Errors: Negative Binomial (error procedures)
Complexity: Negative Binomial (all procedures)
Statements: Exponential (all procedures)

Thus, this result discourages the use of statistical methods that depend on assumptions of normality and encourages the use of non-parametric methods.

Perform a Statistical Analysis

Perform the tests described under Validity Criteria. Sample size is denoted by N.

Discriminative Power

1. Divide the data into two sets: procedures with errors and procedures with no errors. Rank these sets according to their C and S values and perform the Mann-Whitney test to see whether C and S can discriminate between the two sets of procedures (i.e., tell the difference between high quality and low quality software) [Con71, Gib71].

RESULT: The results of the Mann-Whitney test for C are shown in Table 2. The average ranks of C (similar results were obtained for S) for procedures with errors are much higher than the average ranks for procedures with no errors, respectively. We can infer from the low probabilities of higher statistics that C and S for procedures with errors have significantly higher medians in the populations (i.e., that C and S could discriminate a priori between high quality and low quality software) [Con71, Gib71].

Table 2
Mann-Whitney Test: Comparison of Two Samples

<table>
<thead>
<tr>
<th>Sample 1: Complexity - Procedures with errors</th>
<th>Sample 2: Complexity - Procedures with no errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average rank of first group = 85.90, N = 31.</td>
<td>Average rank of second group = 45.24, N = 81.</td>
</tr>
<tr>
<td>Large sample test statistic Z = -6.30</td>
<td>Two-tailed probability of equaling or exceeding Z: 2.95-10</td>
</tr>
<tr>
<td>N: 112 total observations.</td>
<td></td>
</tr>
</tbody>
</table>

2. Divide the data into four categories, as shown in Table 3, according to a critical value of C, C,, so that a chi-square test can be performed to determine whether C, can discriminate between procedures with errors and those with no errors [Con71].

Table 3
Contingency Table

<table>
<thead>
<tr>
<th>Complexity</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>≤ 3</td>
<td>&gt; 3</td>
</tr>
<tr>
<td>No Errors</td>
<td>75</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Errors</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>85</td>
</tr>
</tbody>
</table>

RESULT: The result of the chi-square test is shown in Table 4. From the high value of chi-square and the very small significance level in the samples, we infer that C, could discriminate between procedures with errors (low quality software) and those without errors (high quality software).
Table 4

Summary Statistics for Contingency Tables: $C_\alpha = 3$

<table>
<thead>
<tr>
<th>Chi-square</th>
<th>D.F.</th>
<th>Significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>44.60</td>
<td>1</td>
<td>$2.40 \times 10^{-11}$</td>
</tr>
</tbody>
</table>

$C_\alpha = 3$ will correctly classify 21 out of the 31 procedures with errors (see Table 3). Of course $C_\alpha$ is set no lower than 3 because to do so would cause too many procedures with no errors to be misclassified as procedures with errors (see Table 3).

**Sensitivity Analysis of Critical Value of Complexity**

In order to see how good a discriminator $C_\alpha$ is for this example, we observe the number of misclassifications that result for various values of $C_\alpha$: 1) Type 1 ("error procedures" classified as "no error procedures") and 2) Type 2 ("no error procedures" classified as "error procedures"). As $C_\alpha$ 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 $C_\alpha$ 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 $C_\alpha = 3$, which is the value given by the chi-square test (the chi-square test will not always produce the optimal $C_\alpha$, 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 $C_\alpha$, is multiplied by $C_1/C_2$ ($C_1/C_2 = 1, 2, 3, 4, 5$), 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 a family of five "cost" curves. Naturally, with the higher cost of Type 1 misclassifications taking effect, the optimal $C_\alpha$ (i.e., minimum cost) decreases. However, even at $C_1/C_2 = 5$, $C_\alpha = 3$ is a reasonable choice.

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

4. Perform the Krusal-Wallis test (not shown) to ascertain whether $C$ and $S$ are good discriminators with respect to given values of $E$ (i.e., higher ranks of $C$ and $S$ for higher values of $E$).

**RESULT:** $C$ and $S$ were good discriminators when both procedures with errors and all procedures were evaluated.

**CONCLUSION:** $C$ and $S$ are valid with respect to the discriminative power criterion and either could be used as the initial discriminator during maintenance to distinguish between acceptable ($C \leq 3$, $S \leq 13$) and unacceptable quality ($C > 3$, $S > 13$). However, only one is needed (i.e., $C$ is highly correlated with $S$). Studies [Kho90, Mun89] have shown that a large number of metrics [Li87] can be reduced to a small manageable set that represents the underlying relationship between the quality factor and one or more metrics. It should be noted that it is less expensive to collect $S$ than $C$.

**Tracking**

1. Ideally we want to track a metric against a quality factor over time for a
single component (e.g., procedure). Unfortunately this type of data is not always available because a time history of corresponding quality factor and metric changes is required. This data was not available in this example. In lieu of this data, the chronological sequence of designing and debugging the 31 procedures with errors was used to conduct the tracking test. Corresponding E, C, and S data were available for these modules during the debugging phase. Runs tests were conducted by assigning a "1" if M changed in the same direction as F (i.e., tracks) and a "0" if this was not the case (does not track). The runs test determines whether the binary sequences (runs) are systematic (i.e., M tracks F) or would be expected by chance.

RESULT: The results of the Binary Sequences tests [Sta71] for C is shown in Table 5. C does not track E because the number of 1's and 0's and the number of runs are not statistically different from what we would expect to find in a random sequence. In addition, the Wald-Wolfowitz Runs Test (test for randomness) was performed with the same result [Con71, Gib71] (not shown). The same tests were performed for S with the same result (not shown).

Table 5

Tests for Binary Sequences of Changes in C with changes in E

Element types: 1, 0
Number of 1 elements = 17
Number of 0 elements = 13
Expected number = 15
Statistic of null hypothesis that 1's and 0's are random: Z = 0.54
Probability of equaling or exceeding Z = 0.58
Number of runs = 12
Expected number = 15.73
Statistic of null hypothesis that runs are random: Z = -1.22
Two-tailed probability of equaling or exceeding Z = 0.22

7 APPLYING VALIDATION RESULTS TO MAINTENANCE

During maintenance a control chart is used to identify components that may become difficult to maintain (i.e., excessive change leading to excessive complexity) and which may lead to future unreliable operation. Components whose complexity exceed C0 are flagged for detailed inspection. Also, Table 6 is constructed to show how to allocate resources - people, money, and equipment - to the perfective and corrective maintenance of the components (i.e., project/procedure 2/21 is the highest priority, 4/29 the next highest, etc.), with the restrictions mentioned in the "Introduction". Since C0 = 3, procedures with C0 < 3 receive little or zero priority.

Table 6

PROCEDURES RECEIVING PRIORITY ALLOCATION OF RESOURCES IN MAINTENANCE (MEASUREMENTS MADE DURING DEBUGGING)

<table>
<thead>
<tr>
<th>PROJECT/ERRORS COMPLEXITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROCEDURE</td>
</tr>
<tr>
<td>---------------------------</td>
</tr>
<tr>
<td>2/21 8 16.DECREASING PRIORITY</td>
</tr>
<tr>
<td>4/29 5 13</td>
</tr>
<tr>
<td>2/9 3 8</td>
</tr>
<tr>
<td>4/14 5 8 V</td>
</tr>
<tr>
<td>4/22 1 7</td>
</tr>
<tr>
<td>1/5 5 6</td>
</tr>
<tr>
<td>2/11 3 6</td>
</tr>
<tr>
<td>4/7 1 6</td>
</tr>
<tr>
<td>1/6 2 5</td>
</tr>
<tr>
<td>1/8 1 5</td>
</tr>
<tr>
<td>2/16 2 5</td>
</tr>
<tr>
<td>4/13 1 5</td>
</tr>
<tr>
<td>4/23 1 5</td>
</tr>
<tr>
<td>4/28 2 5</td>
</tr>
<tr>
<td>2/10 1 4</td>
</tr>
<tr>
<td>2/15 1 4</td>
</tr>
<tr>
<td>2/18 1 4</td>
</tr>
<tr>
<td>4/27 3 4</td>
</tr>
<tr>
<td>4/30 1 4</td>
</tr>
<tr>
<td>4/33 2 4</td>
</tr>
<tr>
<td>4/31 1 4</td>
</tr>
</tbody>
</table>

CRITICAL VALUE OF COMPLEXITY = 3
Revalidate Metrics

Repeat the validation tests for C and S as additional metrics data are collected during maintenance, keeping track of the percentage of uses for which the metrics pass (or fail) the validation tests for discriminative power. This statistic provides a measure of the repeatability of the metrics. If the metrics continue to pass the validation tests, using the data aggregated over development and maintenance, continue to use the metrics; otherwise, discontinue using them. Note that the critical values of complexity and statements could change as a result of conducting additional validation tests.

Validate and Apply Metrics in Similar Environments

The final result of the validation exercise is that C and S are valid only with respect to the discriminative power criterion (validated during development) to support the quality control function during maintenance and to provide a rationale for allocating resources to maintenance tasks.

8 SUMMARY AND FUTURE RESEARCH

We described a metrics validation methodology that can be applied to maintenance. The criteria discriminative power and tracking were applied. Non-parametric statistical methods play an important role in evaluating whether metrics satisfy the validity criteria. It was demonstrated that metrics validated during development can be used to establish initial quantitative quality objectives during maintenance and to allocate resources to maintenance tasks. 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 or phase, using our criteria, valid measures of quality on other projects or phases (both similar and different projects and phases)?

REFERENCES


