Abstract

Using our metrics validation methodology, which we developed previously, we show how this methodology can be used across dissimilar multiple projects. We show how to reduce the risk of using metrics on multiple projects. We also show that the choice of metrics and their values can have a significant effect on the quality of software that is achieved and on the cost and amount of inspection that is incurred on multiple projects. A multi-project example, emphasizing the discriminative power validity criterion in support of the quality control function, is presented.

A metrics validation process is defined that integrates quality factors, metrics, and quality functions.

INTRODUCTION

One issue in using software metrics that deserves more discussion in the literature is the need to apply metrics across multiple projects and in a variety of development and operating environments. There are two reasons for this need. One reason is obvious: many organizations develop for or use software in multiple applications and environments. The other reason is not obvious, judging by the paucity of significant discussion of this issue in the literature. This is the need for a validation project V, where metrics are validated, and an application project A, where metrics are applied. We will describe our metrics validation methodology [12] and the issue of dealing with metrics on multiple projects. A second issue is that there should be more recognition of the tradeoff between quality goals and the amount and cost of inspection that is required to achieve those goals, as a function of the metrics set and its values, when selecting metrics for controlling quality. We have addressed these issues by developing a methodology for managing the risks associated with using metrics on multiple projects. By risk we mean the chance that a metric set and its values that are judged to be valid on one project may lead to poor quality or high costs of inspection when applied to another project.

One approach to risk management on software projects is to use measurement to identify high risk parts of the software [11]. Our approach is to assess the risk of using the metrics themselves. When using metrics on multiple projects, we must consider the risk of doing so (i.e., a metric that is valid on one project may not be valid on another project). The reason that metrics may give inconsistent results across projects is that applications (e.g., real-time), product characteristics (e.g., size), process maturity level [10] (e.g., use of inspection), development environments (e.g., design methodology and programming language), personnel (e.g., skill and experience) may be project dependent [2,3,4] and differ considerably among projects. In addition, on a given project, there could be a considerable time lag between the collection of a metric and its validation against a quality factor (see definitions under Framework section) such that the metric no longer is representative of the current condition of a product or process. Of course, one way to reduce risk is to apply metrics on projects with similar characteristics. Since this is not always possible, we must recognize the need to use metrics in multiple diverse environments and to develop other risk minimization techniques. These involve considerations like the characteristics of the samples that are used for validation and application, such as size and type of sample; scaling of metrics across projects; compatibility of statistical methods for validating metrics across projects; costs of using metrics on multiple projects; and inspection requirements necessary to implement metrics on multiple projects.

Counteracting the objective to establish a rational basis for applying metrics are certain myths about what constitutes a "good set of metrics or metrics values". An example is the widespread belief and practice that a specific value of cyclomatic complexity is a "good value" to use on a variety of projects. Actually, the "best value" to use as a discriminator of quality is project dependent, as we have shown [12,13]. It is important that users recognize the quality and cost implications of selecting metrics.

The paper is organized as follows: First, a framework is established which pulls together the concepts and definitions of quality factor, quality metric, validated metric, 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. Then we discuss sampling plan procedures, scaling of metrics across projects, a metrics selection cost sensitivity analysis method, and inspection estimation techniques, which can be used to minimize the risk of using metrics on multiple projects. An example, using the discriminative power validity criterion, shows the application of risk reduction techniques so that a metric, validated on one project, can be applied, with known risk, on a future project to control quality. Lastly, we summarize and comment about future research directions.

FRAMEWORK

The framework of our metrics methodology consists of the following elements, which are keyed to Figure 1: quality factor, quality metric, validated metric, quality function, validity criteria, and metrics validation process. In Figure 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. Each element is defined and described in the following sections.

This diagram is interpreted as follows:

o 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 collected 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).

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

o 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 sections.

Quality Factor
A quality factor F (hereafter referred to as "factor" or "F") is an attribute of software that contributes to its quality [12], where software quality is defined as the degree to which software possesses a desired combination of attributes [12]. For example, reliability (an attribute that contributes to quality) is a factor. A factor can have values, such as the error counts F1, ..., Fn in a set of software components (i.e., an element of a software system, such as module, unit, data or document [12]). We define F to be a type of metric that provides a direct measure of software quality [5]. 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 point 2 and at point 6, respectively, in Figure 1.

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 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) [12]. For example, if there are two components 1 and 2 with M1 = 3 and M2 = 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 [1,5]. 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 Figure 1.

Validated Metric
A validated metric is one whose values have been shown to be statistically associated with corresponding factor values (e.g., M1, ..., Mn have been statistically associated with F1, ..., Fn for a set of software components 1, ..., n) [12]. A validation test of M with respect to F is shown at point 3 in Figure 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 to M whenever it is possible to measure F sufficiently early in the life cycle to permit quality to be controlled (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 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'.

Quality Functions
Quality functions are activities conducted by software organizations for the purpose of achieving project quality goals. Quality functions that our metrics methodology supports have been defined in [12]. The quality function we address in this paper is control.

Quality Control
Quality control is the evaluation of software components against predetermined critical values of metrics (i.e., value of M' which is used to identify software which has unacceptable quality (M') and the identification of components that fall outside quality limits. We denote M' 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' = 3 would be used as a critical value of cyclomatic complexity to discriminate between components that contain errors and those that do not.

Validity Criteria
A set of validity criteria that support our metrics validation methodology have been defined in [12]. The validity criterion we illustrate in this paper is discriminative power. First we define it and then provide an example of its application.
Discriminative Power:
The critical value of a metric $M_i$ must be able to discriminate, for a specified $F_i$, between elements (components $1, 2, ..., n$) of vector $F_i$ [14], in the following way:

$$M_i > M_j < \quad F_i > F_j \quad \text{and} \quad M_i \leq M_j < \quad F_i \leq F_j$$  \hspace{1cm} (1)

for $i = 1, 2, ..., n$, with specified $a$.

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

Would $M_i$ as illustrated in Table 2, partition $F_i$ for a specified $F_i$, as defined in (1)? For example, the data from Table 1 is used in Table 2, with $M_i = 10$ and $F_i = 2$. We see that discriminative power is not perfect in Table 2 (i.e., $F_i \neq 0$). If it is desired to flag components with more than two errors ($F_i > F_j$) for detailed inspection, and if $M_i^* = 10$ (complexity) is validated, it would used on the application project to control quality (i.e., discriminate between acceptable and unacceptable components), as shown in Figure 2. We can use the nominal scale (i.e., component 4 is acceptable, component 5 is unacceptable) [8,9]. The Discriminative Power validity criterion is based on this measurement property.

Since there is seldom a perfect discriminator $M_i$ for $F_i$ (i.e., $O_{12} = 0_{12} = 0$ in Table 2), we use an appropriate statistical method (e.g., chi-square contingency table [6,7]) and representative samples of components to measure the degree to which (1) holds.

RISK OF APPLYING METRICS ON MULTIPLE PROJECTS:

Our metrics validation methodology cautions that to minimize the risk of applying $M_i^*$, $V$ and $A$ should be alike as possible, as mentioned previously [12]. By risk we mean the chance that $M_i^*$, validated at point 3 in Figure 1 and assumed to be valid at point 5, will in fact turn out to be invalid, once there is an attempt to revalidate $M_i^*$ at point 7. However, the reality is that there are many situations in which $V$ and $A$ are dissimilar. Therefore it would be prudent to allow for a variety of projects and to take measures to minimize risk within multiple environments. We use the principle of controlling as many of the non-application specific variables as possible. We give the minimum risk conditions for using the discriminative power validity criterion to control quality.

PROCEDURES TO MINIMIZE THE RISK OF USING METRICS:

Sample Size:
We make $N' = N$, where $N$ is the sample size in $V$ (collected at points 1 and 2 in Figure 1) and $N'$ is the sample size in $A$ (collected at points 4 and 6). It might be argued that proportional sampling should be used (i.e., if there are 1,000 components in $V$ and 10,000 components in $A$ and a 10% sample is used, $N = 100$ and $N' = 1,000$). However, the statistics we use as quality discriminators depend on $N$. In particular, the following exact statistic $T$ (where the chi-square distribution is an approximation) is used to discriminate between high and low quality [7]:

$$T = \frac{N (O_{11}O_{22} - O_{12}O_{21})^2}{n_1n_2 (O_{11} + O_{12})(O_{21} + O_{22})}$$  \hspace{1cm} (2)

where $N$, $n_1$, $n_2$, $O_{11}$, $O_{12}$, $O_{21}$ and $O_{22}$ have the meaning given previously in Table 2. $T$ will be maximized and will provide the greatest discriminative power when $O_{12} = O_{21} = 0$ (no misclassifications); then $T = N$. Therefore, to maintain consistency between $V$ and $A$ and to provide maximum discrimination in $A$, we use $N' = N$.

Minimum Sample Size:
We can estimate the minimum required sample size (estimated at point 1 in Figure 1) for a proportion $q$ of the population range that is included in a sample of size $n$ from the following [7]:

$$n = \frac{(25)(\chi^2_{1-q})(1 + q)/(1 - q)}{.5} \quad \text{with } d.f. = 4$$  \hspace{1cm} (3)

So, if we want an $n$ that will include 90% of the range of values in the population of components on a project, let $q = .9$ and $N' = N \geq n$.

Random Sampling:
Random sampling will preclude bias in selecting components for metrics validation and application. The purpose of this procedure is to be able to make probability statements about the metrics which comprise the sample. Also, this procedure ensures representativeness in the sample. For example, if the population contains a certain proportion of components of a given size then, with equal probability of selection provided by random sampling, these components will have proportional representation in the sample.

A practical exception to the principle of random sampling occurs when the population is so small that the entire population must be used in order to have an adequate "sample" size.

Scaling of $M_i^*$ and $F_i^*$:
When $M_i^*$ is used at point 5 in Figure 1, there is no guarantee that it will be valid in $A$. There are two elements of uncertainty. One is how $F_i^*$ will partition the data between $O'_{11}$, $O'_{12}$ and $O'_{21}$, $O'_{22}$ (see Table 3). The second is how $M_i^*$ will partition the data between $O'_{11}$, $O'_{21}$ and $O'_{12}$, $O'_{22}$. We deal with these elements as follows:

Element 1: Since $F_i^*$ cannot be collected until point 6, we don't know at point 5 how $F_i^*$ will partition the data. We will make the assumption that the proportion of components in the population that have errors will not differ significantly between $V$ and $A$. That is, we assume $n' = n_1, n' = n_2$ for $N' = N$ (see Table 3). Therefore, to make the samples comparable
between \( V \) and \( A \), we set \( F' = F \) and \( N' = N \), as previously stated (see Table 3).

**Element 2:** Since we set \( F' = F \) and we know that \( M' \) partitions the data between \( N'_1 \) and \( N'_2 \) (see Table 3), we want assurance that \( O'_{21} \geq O_{21} \) and \( O'_{21} \leq O_{21} \). That is, we don't want correct classifications \( O'_{22} \) to decrease and Type 1 misclassifications \( O'_{22} \) to increase in going from \( V \) to \( A \) (these are the classifications associated with low quality). We want to flag as many low quality components as possible (see Table 3).

In the process of providing this assurance, correct classification \( O'_{11} \) may decrease and Type 2 misclassifications \( O'_{12} \) may increase. However this is not as serious as having Type 1 increase, because Type 2 is associated with high quality components. Therefore the following hypothesis test \( H_s \), corresponding to the data partition \( M' \subseteq M'' \), is appropriate [7]:

\[
H_s: N'_1 > N_1 \quad \text{(if true, no assurance that } O'_{21} \leq O_{21}) \\
H_s: N'_1 \leq N_1 \quad \text{(if true, a degree of assurance that } O'_{21} \leq O_{21})
\]

If we are able to reject \( H_s \) and accept \( H_s \), using the assumption stated in **Element 1**, we will have a degree (we say degree because we won't know until point 6 of Figure 1 how \( F' \), will partition the data) of assurance that the proportion of Type 1 misclassifications in the population of components is no greater in \( A \) than in \( V \); thus it would be appropriate to use \( M' \) in \( A \). On the other hand, if we accept \( H_s \), we have the option of not using \( M''_s \), or of using it and realizing that it is possible that \( M' \) will turn out to be (at point 7) a higher value than is desired in \( A \).

Table 4 shows the relevant quantities for the hypothesis test, where the classifications \( N_1, N_2, \) corresponding to \( V \), \( N'_1, N'_2 \), corresponding to \( A \), are shown in the contingency table [7]:

1. If \( N'_1 > N_1 \), accept \( H_s \) (5)
2. Otherwise, compute \( T \) (equation 2), using the variables shown in Table 4.
3. If \( T > \chi^2_p \) with d.f. = 1, accept \( H_s \)
4. Otherwise, accept \( H_s \)

It is important to note that the sample \( N' \) is collected and the above analysis is performed at point 4 in Figure 1 prior to collecting \( M' \) and comparing it with \( M'' \), for the purpose of controlling quality at point 5. We wish to assess the risk of using \( M' \) prior to making a commitment to applying it to the population of components \( N'' \), at point 5. We can consider the use of the sample as a metrics prototyping process.

**Cost Sensitivity Analysis of Critical Value of Complexity:**

Another risk minimization technique involves an analysis of the sensitivity of \( M \) to the costs of misclassification, when the discriminative power validity criterion is applied. This analysis is performed at point 3 in Figure 1. In order to illustrate the procedure, we use \( C_s \), cyclomatic complexity for \( M \) and \( E_s \) error count, for \( F \). In this example, we observe the number of misclassifications that result for various values of \( C_s \) 1) Type 1 ("error components" classified as "no error components") and 2) Type 2 ("no error components" classified as "error components"). This is shown in Table 3 and Figure 3. As \( C \) increases, Type 1 misclassifications increase because an increasing number of high complexity components, many of which have errors, are classified as having "no errors". Conversely, as \( C_s \) decreases, Type 2 misclassifications increase because an increasing number of low complexity components, 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 \( C_s = 3 \); this value can be obtained from a chi-square contingency table, as described previously.

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 \( C_s \) 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 Figure 4. Naturally, with the higher cost of Type 1 misclassification taking effect, the optimal \( C_s \) (i.e., minimum cost) decreases.

**Estimate of Inspection:**

Another consequence of misclassifications is the amount of inspection required. Total inspection (Type 2 misclassifications + error components correctly classified) is given by (6) and wasted inspection (Type 2 misclassifications) is given by (7) and are estimated at point 3 in Figure 1:

\[
I_s = \frac{O_{11} + O_{21}}{N} = N_f/N
\]

\[
I_w = O_{12}/N
\]

Total inspection can be re-estimated during metrics prototyping at point 4, Figure 1, as given by (8), once \( N'_1 \) is known (see Table 4):

\[
I'_s = N'/N
\]

**Summary:**

We summarize the procedures involved in minimizing the risk of applying the discriminative power validity criterion on multiple projects:

1. Make sample sizes \( N' = N \).
2. Estimate minimum sample size \( n \), where \( N' = N > n \).
3. Draw random samples of size \( N' = N \).
4. Set \( F' = F \).
5. Perform a hypothesis test \( H_s \) on \( N'_1 \), corresponding to \( M' \subseteq M'' \), to estimate the likelihood of \( M' \) being too large in \( A \).
6. Perform a cost sensitivity analysis on \( M \) to find the optimal \( M \) over a range of cost assumptions.
APPlying Validation RESULTS TO A Future project

In this example we use projects 1, 2 and 3 (N = 43) from Table 5 for V and a random sample subset of project 4 (N′ = 43) from Table 5 for A. The data used in this example were collected from actual software projects. The purpose of this validation is to determine whether cyclomatic number (complexity (C)), validated in V with the discriminative power validity test, could be used to control the factor reliability, as represented by the factor error count (E) in A.

In performing the discriminative power validation, we are interested in answering the following question: How well will C, validated in V work as a discriminator of quality in A? Using the procedures developed under "Procedures to Minimize the Risk of Using Metrics", we perform the analysis as follows:

Sample Size: Set N′ = N.

Minimum Sample Size: Using equation (3) with q = .9 and α = .05, gives n = 46. Since the population size of Projects 1,2,3 is 43 components, we set N′ = N = 43. N = 43 provides a q = .89 at α = .05. Actually, a random sample, N′ = 43, for project 4 includes 97.7% (q = .977) of the range of C in the population of 69 components.

Random Sampling: In order to obtain an adequate N, the entire population (N = 43) is used. N′ = 43 is obtained by random sample.

Scaling of C′, and E′,: Set E′ = E, = 0. Using equations (4,5) and Table 6 (where C′ = 1 partitions V and A as shown) perform the following hypothesis test:

H₀: N′ > N₁
H₁: N′ ≤ N₁

Since N′ > N₁, don’t accept H₀ at this stage.

Compute T (equation 2) = .748.

Since T < χ²₀.05, with α = .05 and d.f. = 1, reject H₀ and accept H₁.

In general, this outcome could imply that Type 1 misclassifications would increase in A. However, with C₁ = 1, the lowest possible value, (see below), this won’t happen in this case, so we would apply C₁ = 1 to the population of components.

Cost Sensitivity Analysis of critical Value of Complexity:

We see from Table 7 that the optimal C₁ = 1, where maximum chi-square occurs (i.e., maximum discriminative power) and where minimum α (statistically significant value) occurs. Also, C₁ = 1 is optimal for various cost ratios C₁/C₂, including C₁/C₂ = 5. This is shown graphically in Figures 5 and 6 for V. In Figure 5, Quality, as represented by Errors Found, and Cost (C₁/C₂ = 5) are plotted against C₁ (see Table 7). It is seen that highest quality and lowest cost is obtained at C₁ = 1. In Figure 6, Misclassifications (Type 1 + Type 2) and Discriminative Power (Chi Square) are plotted as a function of C₁ for V (see Table 7). It is seen that chi square acts as an excellent discriminator of quality: when it increases, misclassifications decrease and when it decreases, misclassifications increase. Minimum misclassifications occurs at C₁ = 1.

estimate Inspection:

Using equations (6, 7), we estimate total and wasted inspection during metrics validation and using (8), we re-estimate total inspection during metrics prototyping. This relationship is shown graphically in Figure 7, where Inspection (total) and Errors (remaining errors/KLOC) are plotted as a function of C₁ for V (see Table 7). The remaining errors are those that were not found as a result of using the given C₁. It is seen that C₁ = 1 yields minimum remaining errors but requires the most inspection.

Postmortem Analysis

Now, in Project 4, during the design phase, we use C₁ = 1 to control quality. To see how well C₁ = 1 fares against Project 4 data, we provide Table 8. For comparison purposes, this table shows the following: the validation of C₁ = 1 in V and the performance of C₁ = 1, as if it had been applied in V; the application of C₂ = 1 in A; and a postmortem analysis of A, after collecting E′. We perform a postmortem analysis of C’ with E’ for Project 4 to see what would have happened at point 5 in Figure 1 if we had known at point 5 what we eventually learned at point 7. Table 8 indicates that we would have chosen C₂ = 3, rather than 1, if we had known E’ at point 4 in Figure 1. We note in Table 5 that A has less than half the error density of V (possibly due to increased programmer experience and learning). This contributes to the difference in C₂ (i.e., C₂ can be larger in A to achieve roughly the same quality as in V).

We see from Table 8 that the application of C₂ = 1 in A does better than estimated in V with respect to Type 1 misclassifications (none), but worse with respect to Type 2, cost, and inspection. We also see that C₂ = 1 does better than C₂ = 3 in catching errors (i.e., zero Type 1 misclassifications) but worse with respect to Type 2, cost, and inspection. Curves similar to those shown in Figures 4-6 were obtained for A (not shown). In retrospect, C₂ = 1 was a good choice for achieving high quality software but its use resulted in much unnecessary inspection. These are consequences of using a low value of C’.

It is seen that the optimal value of C’ is project dependent.

Summary and Future Research

We described and illustrated a metrics validation methodology that has validity criteria, which support the quality control function. The validity criterion discriminative power was defined and illustrated. 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 how the discriminative power validation criterion can be used to validate a metric on one project and apply it on another project. We discussed sampling plan procedures, scaling of metrics across projects, a metrics selection cost sensitivity analysis method, and inspection estimation techniques that minimize the risk of using metrics on multiple projects. Finally, we showed how to deal with the important tradeoff relationship involving quality, inspection and metrics selection across multiple projects.

Future research is needed to extend the methodology by identifying risk reduction procedures that can be applied to the other validity criteria of our metrics validation methodology [12].

Acknowledgements

We acknowledge the support provided for this research project by the Naval Surface Warfare Center and the Army Operational Test and Evaluation Center.

REFERENCES


Unacceptable Components

Acceptable Components

Application Project Time

Figure 2. Application of Metrics to Quality Control (discriminative power) for Components 1, 2, ..., n

Figure 1. Metrics Validation Process

Figure 3. Validation Project 1 Time Line
1. Collect M
2. Collect P
3. Validate M with P

Figure 4. Validation Project 2 Time Line
4. Collect M'
5. Collect P'
6. Assess, Control, Predict

Figure 6. Validation Project: Misclass. & Discrim. PMR. vs. Complexity

Figure 7. Validation Project: Inspection & Errors vs. Complexity
Figure 3

Figure 4
### Table 1
(Validation Project)

<table>
<thead>
<tr>
<th>Component</th>
<th>N</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>7</td>
</tr>
</tbody>
</table>

### Table 2
(Validation Project)

\[
\begin{array}{ccc}
| F' \leq F' | O_{11} & 1 & O_{12} = 0 \\
| F' > F'  | O_{21} = 1 & O_{22} = 2 \\
\end{array}
\]

\[N = n_1 + n_2\]

\[O_{ij} = \text{count of observations in cell } i,j.\]

\[O_{11}, O_{22}: \text{correct classifications.}\]

\[O_{12}, O_{21}: \text{incorrect classifications.}\]

### Table 3
(Validation Project Contingency Table)

<table>
<thead>
<tr>
<th>N ≤ N'</th>
<th>N &gt; N'</th>
</tr>
</thead>
<tbody>
<tr>
<td>F' ≤ F'</td>
<td>O_{11}</td>
</tr>
<tr>
<td>F' &gt; F'</td>
<td>O_{21}</td>
</tr>
</tbody>
</table>

\[N = n_1 + n_2\]

### Table 3 (Continued)
(Application Project Contingency Table)

<table>
<thead>
<tr>
<th>N' ≤ N''</th>
<th>N' &gt; N''</th>
</tr>
</thead>
<tbody>
<tr>
<td>F'' ≤ F''</td>
<td>O'_{11}</td>
</tr>
<tr>
<td>F'' &gt; F''</td>
<td>O'_{21}</td>
</tr>
</tbody>
</table>

\[N' = N, F' = F, N'' = N'\]

\[O'_{ij}, O'_{11}, O'_{12}, O'_{21}, O'_{22}: \text{correct classifications.}\]

\[O_{12}, O'_{21}, O'_{12}, O'_{22}: \text{incorrect classifications.}\]

### Table 4
(Continuity Table for V and A Project Comparisons)

<table>
<thead>
<tr>
<th>V</th>
<th>N,</th>
<th>N'</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>N',</td>
<td>N''</td>
</tr>
</tbody>
</table>

### Table 5
(Project Application Components Statements Errors (with errors))

<table>
<thead>
<tr>
<th>Components</th>
<th>Statements</th>
<th>Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>String Processing</td>
<td>11 (5)</td>
<td>136 10</td>
</tr>
<tr>
<td>Directed Graph Analysis</td>
<td>31 (12)</td>
<td>430 27</td>
</tr>
<tr>
<td>Directed Graph Analysis</td>
<td>1 (1)</td>
<td>13 1</td>
</tr>
<tr>
<td>Data Base Management</td>
<td>69 (13)</td>
<td>1021 26</td>
</tr>
</tbody>
</table>

Number of components: 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.

### Table 6
(Continuity Table for V and A Project Comparisons)

<table>
<thead>
<tr>
<th>V</th>
<th>N, = 25</th>
<th>N' = 18</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>N' = 21</td>
<td>N'' = 22</td>
</tr>
</tbody>
</table>

\[N = 43\]

### Table 7
(Validation Project Projects 1, 2 and 3)

<table>
<thead>
<tr>
<th>Components</th>
<th>25 with no errors, 18 with errors (38 errors total)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C, (x^2)</td>
<td>(19.00) (1.288E-5) (35) (5)</td>
</tr>
<tr>
<td>2</td>
<td>(9.52) (2.03E-3) (27) (19)</td>
</tr>
<tr>
<td>3</td>
<td>(15.12) (1.000E-4) (27) (19)</td>
</tr>
<tr>
<td>4</td>
<td>(8.93) (2.80E-3) (24) (24)</td>
</tr>
</tbody>
</table>
### Table 8

**Applying Validation Results to a Future Project**

<table>
<thead>
<tr>
<th>Function</th>
<th>Validate</th>
<th>Apply</th>
<th>Postmortem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project</td>
<td>1, 2, 3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Phase</td>
<td>Design, Test</td>
<td>Design</td>
<td>Test</td>
</tr>
<tr>
<td>Components</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>43</td>
<td>69</td>
<td>69</td>
</tr>
<tr>
<td>No Error/Error</td>
<td>25/18</td>
<td>56/13</td>
<td>56/13</td>
</tr>
<tr>
<td>$C_c$</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Components Correctly</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Classified</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No Error</td>
<td>22</td>
<td>28</td>
<td>50</td>
</tr>
<tr>
<td>Error</td>
<td>15</td>
<td>13</td>
<td>11</td>
</tr>
<tr>
<td>Components Misclassified</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type 1</td>
<td>3</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Type 2</td>
<td>3</td>
<td>28</td>
<td>6</td>
</tr>
<tr>
<td>Cost ($C_1/C_2 = 5$)</td>
<td>18</td>
<td>28</td>
<td>16</td>
</tr>
<tr>
<td>Total Inspection</td>
<td>41.8% (51.2%)'</td>
<td>59.4%</td>
<td>24.6%</td>
</tr>
<tr>
<td>Wasted Inspection</td>
<td>7.0%</td>
<td>40.6%</td>
<td>8.7%</td>
</tr>
</tbody>
</table>

* Re-estimated during metrics prototyping in A.