A Classification Procedure for the Effective Management of Changes during the Maintenance Process

53-61

136138

Lionel C. Briand and Victor R. Basili [1]

Computer Science Department and Institute for Advanced Computer Studies
University of Maryland
College Park, MD, 20742

N93-17169

P-12

To be published in the proceedings of the IEEE Conference on software maintenance, Orlando, Florida, USA, November 1992.

Abstract

During software operation, maintainers are often faced with numerous change requests. Given available resources such as effort and calendar time, changes, if approved, have to be planned to fit within budget and schedule constraints. In this paper, we address the issue of assessing the difficulty of a change based on known or predictable data. This paper should be considered as a first step towards the construction of customized economic models for maintainers. In it, we propose a modeling approach, based on regular statistical techniques, that can be used in a variety of software maintenance environments. This approach can be easily automated, and is simple for people with limited statistical experience to use. Moreover, it deals effectively with the uncertainty usually associated with both model inputs and outputs. The modeling approach is validated on a data set provided by the NASA Goddard Space Flight Center which shows it has been effective in classifying changes with respect to the effort involved in implementing them. Other advantages of the approach are discussed along with additional steps to improve the results.

Key words: maintenance process, change difficulty, change request management.

[1] Research this study was supported in part by NASA grant NSG 5123 and the Vitro Corporation (IAP Member)

10005788L

1 Introduction

Given the limited resources (i.e. effort and calendar time) available to the maintenance activity within software organizations and the number of change requests proposed, difficult decisions need to be made. These decisions include: which changes to implement, how much optional functionality to provide in enhancements. A large amount of total software effort is spent on maintenance [LS80, GRA87]. Changes in the form of corrections, enhancements or adaptations effect the software source code and/or the documentation. Some of these changes are crucial, others are less important. Therefore, when one considers the global cost and variety of maintenance activities, management of changes becomes an important and complex task. It requires the support of models so we may perform systematic comparison of the costs and benefits of changes before implementing them [RUV92]. One approach is to build such models based upon past project experiences.

To this end, effort models have to be designed to predict resource usage and optimize the cost-effectiveness of the maintenance process. Well defined modeling procedures need to be established so they can be repeated and refined, allowing the model to evolve consistently as new data are collected.

This paper describes a modeling procedure for constructing a predictive effort model for changes during the maintenance phase. This technique is intended to handle small data sets and the uncertainty (i.e. for cost or technical reasons) usually associated with model inputs and outputs (i.e. is this particular predication believable?). We assess the feasibility of building such a model using a data set that describes several projects in the SEL environment at the NASA Goddard Space Flight Center. Based upon the results of the analysis, we also make recommendations for improving the data collection process.

2 Context of Study and Experiment Design

In this study, we use a data set consisting of 163 changes collected on four different maintenance projects. Each change is represented by a vector consisting of a variety of metrics associated with the change. The four projects are referred to in the paper as projects p1, p2, p3, p4. These projects are from the same application domain: satellite ground support software written in FORTRAN.

The change process in the SEL environment has two main phases: an "understanding" phase where the change is determined and isolated in the system and an "implementation" phase, where the change is designed, implemented and tested.

The effort associated with both the understanding and implementation phases is collected on discrete scales (i.e. ordinal) in order to facilitate the data collection from a maintainer's perspective. The effort range is divided into five intervals: below one hour, between one hour and one day, between one day and one week, between one week and one month, above one month. For each change performed, the appropriate understanding effort and implementation effort intervals are recorded by the person making the change. These effort intervals are indexed from 1 to 5 and will be referred to as *difficulty indices* in the paper.

All the change-related data used in this paper was collected on a standard form (see Appendix). The metrics collected range from measures on a continuous scales (e.g., number of components added, number of lines of code added) to categorical measures (e.g., source of the change, technical description of the change). Some of these metrics are predictable before starting the design of the change, others can only be assessed after the implementation of the change has begun.

In this paper, we focus exclusively on the effort spent to implement (i.e design, code, test) a change. There are two reasons for this: 1) Almost no information is available to the maintainer before the understanding phase. Therefore, no prediction model can be built. 2) In this environment, the effort expended in the understanding phase is generally somewhat smaller than the effort expended during the implementation. It is thus more essential to use a predictive model for the implementation phase.

The available metrics are defined as follows:

- Type of modification (correction, enhancement, adaptation).
- Origin of the error in the software life cycle. This is referred to as *source* in the text (requirements, specifications, design, code, previous change).
- Software products effected by the change (code only, code and design). This is referred to as *objects* in the text.
- Number of components added, changed, deleted. They are referred to as *comp.add, comp.ch., comp.del.*, respectively.
- Number of lines of code added, changed, deleted. They are referred to as *loc. add., loc. ch., loc. del.*, respectively.
- Change technical description (initialization, logic/control structure, user interface, module interface, data structures, computational) . This metric is referred to as *ch.desc.*

During the *understanding phase*, estimates can be made of the first three metrics. The number of components involved in a change can also be approximated since the change is isolated in the system architecture. But any prediction in terms of lines of code to be added, deleted or changed is still complex at this point and can only be predicted at a coarse level of precision.

3 The Modeling Approach

Considering the discrete nature of the effort data reported during maintenance, the prediction issue becomes a classification issue, i.e. in which effort class will the change probably lie? The maintainer can only predict values for most input metrics with a certain degree of uncertainty. It is important that the modeling process takes this constraint into account. This help to make the generated model easy to use. Also, our data set is small and contains discrete explanatory variables. Therefore, we need a modeling approach which is both effective on small samples and which handles discrete and continuous explanatory variables in a consistent way.

3.1 The Modeling Process Steps

A high level view of the model construction process can be defined as follows:

1- *Identify Predictable Metrics*. Identify the metrics, among those available, that are predictable before the implementation phase. For ratio and interval metrics that are predictable early but only with a certain degree of uncertainty, the range is recoded as an ordinal range with a set of ordered classes. These classes reflect a reasonable level of prediction granularity. For example, a ratio level metric range like "number of components added" could be divided into three intervals forming the three metric classes low, average, and high.

2- *Identify Significant Predictable Metrics*. Identify a subset of the predicable metrics that appear to be good predictors of the difficulty index, using a consistent evaluation technique for all candidates.

3- *Generate a Classification Function.* Associate the resulting metrics in a classification function which has the following form:

Predicted_Difficulty = Classification_Function (Significant_Predictable_Metrics)

where Predicted_Difficulty = some classification scheme based on the difficulty indices, e.g., {easy, difficult} and Significant_Predictable_Metrics = {some of the predictable metrics collected on the Maintenance Change Report Form which appear as good predictors}

4- *Validate the Model.* Conduct an experiment on a representative (i.e. in terms of size and quality) set of data. Two measures that can be used to validate the model are: Average Classification Correctness (i.e. ratio of number of correct classification / total number of performed classifications), and Indecision Rate (i.e. ratio of number of undecidable cases / total number of changes to be classified). The latter reflects the need for such a model to deal with output uncertainty, therefore warning the user whenever a certain level of confidence is not reached for a specific prediction.

3.2 An Implementation of the Modeling Process

This section presents a possible implementation of the previously described process. Our goal in defining such a procedure can be described by the following points:

• We want the generated model to be as simple to use as possible.

• The uncertainty associated with the model inputs at the time of prediction must be taken into account by the model, i.e., intervals rather than values should be used as model inputs.

• The model should be able to provide some estimated risk of error associated with each classification. Thus, the user would be able to select a minimal level of confidence (i.e. maximum risk) that would differentiate the model classifications as believable or non-believable.

• The steps of the procedure are:

1- *Identify Predictable Metrics.* The input is a set of available metrics. The output is a set of metrics whose values are either known or predictable, with a certain degree of accuracy, before the change implementation phase.
There are several processes for selecting the set of predictable metrics. The determination of predictability can be either based on interviews with people with a good knowledge of the maintenance process (and then refined with experience) or observed through controlled experiments [BSP83, BW84]. Both help to determine the average estimation accuracy that can be reasonably expected for a given metrics.

The range of each continuous / ordinal predictable metric is divided into intervals (e.g., percentiles, natural clusters [DIL84]). The more accurately predictable the metric, the more numerous and narrow the intervals can be. We recode the metric ranges according to their respective predictability so the maintainer can easily select the right interval and use some of the predictive power of metrics not measurable before the *implementation phase*. These intervals are called *metric classes* in the paper.

Our need to define these metric classes for predictable metrics stems from the impossibility of relying exclusively on measurable (at the time of prediction) metrics, e.g. building an accurate model for predicting change effort is likely to require measures of change size that are not available before the implementation phase. We have no choice other than taking into consideration metrics that cannot be measured but only approximated with a certain degree of precision by the maintainer after the *understanding phase* of the change process.

2- *Identify Significant Predictable Metrics*. The input to the second step is the set of predictable metrics from the first step and the outputs are a subset of significant predictors and their corresponding association table. This association table distributes the difficulty indices across the metric classes defined on each predictor value domain.

Consider as an example Table 1 which shows the association table of the metric *number of lines of code added* across the four difficulty classes (class 5 has so few changes that we merge it to class 4). This table is calculated based on the actual distributions in the data set considered for modeling. Each column represents a metric class (e.g. > 30 implies that the number of loc added is more than 30) and each row an index of difficulty. With respect to each predictable metric and using its calculated distribution of difficulty indices, an average difficulty index (i.e ADI) is calculated for each metric class (shown in the bottom row of Table 1). The calculation of a meaningful and statistically significant ADI requires us to set up the metric classes in a way that guarantees a minimum number of changes in each of them.

	Loc added		
DI	< 10	[10 30]	> 30
1	7%	0%	0%
2	48%	36%	9.5%
3	42%	60%	40.5%
4	3%	4%	50%
ADI	2.40	2.68	3.40

Table 1: "number of lines of code added" distribution

Taking the association table *Table 1* as an example, the calculated index averages look consistent with what was expected. The ADI seems to increase substantially with the number of lines of code added. In general, with respect to the ratio and interval level metrics whose the value domains have been recoded in successive metric classes (see step 1), significant differences should exist between class ADIs. Based on a F-test, a one-way analysis of variance (ANOVA) can be performed and the statistical level of significance of the metric class ADI differences may be estimated [CAP88]. Whenever the 0.05 level of significance is not reached, the boundaries should be recoded in a way that minimizes the level of significance. Since all the continuous metric ranges have been recoded into an ordinal scale, we have to calculate the degree of association between the difficulty indices and the metric classes in order to assess the predictive power of each metric. One approach consists of computing the Chi-Square statistic (which is valid at the nominal level [CAP88]) for each metric association table. A statistical level of significance characterizing the association between the difficulty indices and the metric classes is calculated based on the generated Chi-square value. Thus, the top ranked metrics showing sufficient degree of association are selected as parameters potentially usable to build a multivariate prediction model. Some more sophisticated measures of association (i.e. PRE-measures of associations [CAP88]) can provide more intuition/information about the associations and therefore allow an easier selection. However, this issue is beyond the scope of this paper.

3- *Generate a Classification Function*. The input to the third step is the set of association tables of significant predictable metrics and the output is a classification model that predicts an expected difficulty index associated with changes. Note that although five difficulty indices are defined on the change form, a small minority of the changes (5%) actually lie in the extreme intervals (i.e. intervals 1,5).

This makes classification into these intervals extremely difficult. Also, since 80% of the chances belong to classes 2 and 3, we will first build a classification model intended to differentiate these two classes: less than one day (i.e. referred as easy), more than one day (i.e. referred as difficult). In section 4.2, we will refine our classification by dealing with a "more than one week" class (i.e. indices 4 and 5). Thus, based on the generated classifications the user will be able to make decisions with respect to the requested implementations of changes. This is done by comparing the predicted difficulty to both the available resources and the expected gains.

The process of building a classification function is composed of two steps:

1- Perform a regression: Based on all the available association tables and the corresponding ADIs for each change in the data set, we perform a stepwise linear regression [DIL84] of the following form:

Actual_difficulty_index = W1 * ADI_metric1 + ... + WN * ADI_metricN

Due to interdependencies between metrics, only a subset of the preselected metrics remains in the generated prediction function (i.e. only the one showing, based on a F-partial test, a level of significance below 0.05). In order to make the model easier and less costly to use, the number of parameters in the regression equation can be minimized. In this case, one or several parameters are removed (especially when they show a statistical significance close to 0.05) and the resulting models are evaluated. Then, the user has to assess the loss of correlation against the ease of use gained by removing parameters from the model. If the tradeoff appears reasonable, then the new model is adopted. Weights are calculated for each remaining parameters and the resulting optimized linear function allows us to calculate an difficulty index expected value. This may be used to classify the change based on the realistic assumption that: the closer the expected value of the difficulty index to an actual difficulty index, the more likely the corresponding change belongs to the matching effort class. Therefore the following interval-based decision rule is used to make classifications.

2- Define a decision rule for classification: the predicted difficulty index range is divided into three intervals (i.e. easy change predicted, undecidable, difficult change predicted) in a way that guarantees a maximal average classification correctness. For example, the boundaries for classifying a change as either less or more than one work day can be defined as in Figure 1. The classification of future changes will be performed according to the interval in which their calculated difficulty index will lie.

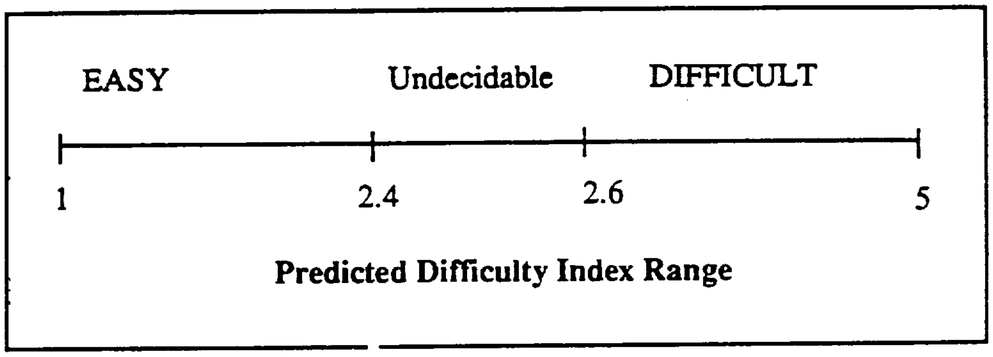

Figure 1 : Example of decision intervals

The process for creating these decision boundaries can be described as follows:

1- The user defines a *risk / loss function* having the following form:

$$Expected\_loss = Weight1*MR1 + Weight2*MR2$$

where MRn is the misclassification rate calculated for changes actually in class n.

The loss function weights can be defined according to the respective costs of misclassification associated with each class. Most of the time, this weight will be set to one. A search algorithm can then be used to determine the interval between two neighboring changes on the predicted index range that provides the best decision boundaries (i.e. that minimizes the risk / loss function). These two neighboring changes form the boundaries of the smallest possible *undecidable interval* on the range.

2- In a stepwise manner, this interval can be widened on both sides of the index range according to some automatable process. For instance, the interval can be expanded in a stepwise manner, including one more change at a time on each side of the interval, until a maximal expected loss value (i.e. predefined by the user) is reached. Based on this process, the user will be able to determine the boundaries of the decidable intervals corresponding to the desired level of risk.

4 A Validation Study

According to the procedure defined above and based upon the previously described four project data set, the significance of each available metric as a predictor is assessed. Table 2 shows the Chi-square-based levels of Significance. Then, in order to build the needed classification models, the metrics yielding a good level of significance are selected. First, we build a general model usable for any project in the same environment. This model is intended to be useful at the start of a maintenance process when not enough data are available to create a project specific model.
Then, we build an independent classification model for each project which is expected to be more accurate with respect to future changes for each specific system, respectively. The various results will be compared in order to assess the validity of cross-project models in this environment. The ranges of the continuous metrics were recoded according to the previously described procedure. Two or three metric classes were defined for each of the metrics, according to the predictability level of the metric and the distribution of the changes on their respective range. In other words, the interval boundaries were chosen in a way that reflected their predicability, optimized the classification power of the metric (i.e. optimized the chi-square) and guaranteed, to the extent possible, a sufficient number of changes within each metric class.

Metric	Level of significance	Metric classes
Type	0.004	Correction, enhancement, adaptation
Source	0.0000	Requirements, specifications, design, code, previous change
Ch.desc.	0.0000	initialization, logic, interface, data structure, computational
Loc. add.	0.0000	<10, [10, 30] , >30
Loc. Ch.	0.0000	<10, [10, 25] , >25
Loc. Del.	0.0006	<2, [2, 15] , >15
Comp. add.	0.0002	0, > 0
Comp. ch.	0.0000	<2, [2, 5] , >5
Objects	0.0005	code only, code and design

Table 2: Level of significance and class boundaries / categories of metrics

4.1 A General Model

This model is intended to be specific to the NASA SEL environment. It has been built based on systems belonging to the same application domain and therefore may not represent necessarily other domains accurately. Table 3 shows for each selected metric, the class ADIs and the corresponding result of the one way analysis of variance [CAP88] that assessed the statistical significance of the ADI variations across metric classes. They all appear below 0.05 and we can therefore say that the metric classes with respect to continuous metrics have been adequately defined because they show significant ADI differences.

Table 4 shows two distinct regression-based classification functions (PI stands for Predicted difficulty Index). Note that the parameters of the regression equations are the metric association table-based ADIs and not the metric values themselves. For the sake of simplification, the names of the metrics are shown in the equations. For each function, the calculated regression equations are given with the respective level of significance of each metric (i.e.shown between brackets above the equations and based on partial F-tests).

If the metric does not appear significant at a 0.05 level, then they are excluded of the equation. The global coefficient of determination R^2 is also given. The first one was obtained by performing a stepwise regression using the class ADIs of the significant predictable metrics. Only one of the lines of code (i.e. loc) based metrics was retained in the equation: *loc.ch*. Then, in an attempt to avoid the use of this metric (i.e. which is still the most difficult to assess despite the coarse defined metric classes), we recalculated the equation parameters when ignoring it. The coefficient of correlation did not appear much affected by the change. This can be explained by the higher significance of the remaining parameters and their stronger calculated coefficients that show a strong interdependence with *loc.ch*. In other words, they partially compensated the loss of explanatory power due to the removal of *loc.ch*.. Thus, the generated model becomes even easier to use and does not loose much of its accuracy (see Table 5).

Metrics	Level of significance	ADIs for each category
Type	0.003	[2.56 , 2.36, 2.95]
Source	0.0000	[3.04, 3.29 2.42, 2.33, 2.27]
Ch.desc.	0.0000	[2.2, 2.9, 3.0, 3.1, 2.4, 2.9, 2.8]
Loc. add.	0.0000	[2.4, 2.68, 3.4]
Loc. Ch.	0.0000	[2.4, 2.8, 3.14]
Loc. Del.	0.0001	[2.6, 2.8, 3.4]
Comp. add.	0.0000	[2.64, 3.63]
Comp. ch.	0.0000	[2.41, 3.0, 3.31]
Objects	0.01	[2.63, 3.03]

Table 3: Metric class ADIs

	Description of the Models	R-sq
Model 1	$\underset{}{P_I} = -\overset{(0.0)}{4.22} + \overset{(0.0)}{0.59}\ \text{Source} + \overset{(0.0)}{0.62}\ \text{Ch.desc} + \overset{(0.0008)}{0.58}\ \text{loc.ch} + \overset{(0.04)}{0.38}\ \text{Comp.add} + \overset{(0.004)}{0.36}\ \text{Comp.ch}$	0.50
Model 2	$P_I = -\overset{(0.0)}{3.95} + \overset{(0.0)}{0.68}\ \text{Source} + \overset{(0.0)}{0.69}\ \text{Ch.desc} + \overset{(0.01)}{0.49}\ \text{Comp.add} + \overset{(0.0)}{0.56}\ \text{Comp.ch}$	0.46

Table 4: General models

Table 5 shows the *classification correctness* (i.e. rate of correct classification) obtained when using the above models (Table 4). The decision boundaries have been optimized to yield the best results. First, they have been selected to yield a 0% indecision rate (column IR = 0% in Table 5). Then the undecidable interval has been widened in order to demonstrate the possibility of selecting decision intervals that fit the user's need in terms of classification correctness (column IR > 0% in Table 5). In this case, the selected interval boundaries are arbitrary and are shown for the sake of example. The row "classification" indicates the classification performed (i.e. easy changes = [1-2] or [1-3]). Each cell contains, for all models, the undecidable interval boundaries between brackets and the corresponding classification correctness. Whenever the undecidable interval has been widened (i.e. IR > 0%), the corresponding indecision rate is given.

Despite the mediocre coefficient of determination, a particularly good correctness has been obtained when the interval [1-3] represents easy changes. However, the results appear much less satisfactory for the other classification performed. Nonetheless, this can be substantially improved by widening the undecidable interval. Thus, the model appears usable for at least a subset of the changes. However, when possible (i.e. enough project data are available), project specific models should be used as demonstrated in the next paragraphs.

4.2 Project Specific Models

Table 6 shows optimal equations resulting from stepwise regressions performed independently for each of the four projects. The format used is the same as in Table 5. Differences between models are observable with respect to the variables selected. This does not necessarily mean a real variation in the impact of the explanatory variables across projects. It may be due to a lack of variation of a variable within a project specific data set.

	Description of the Models	R-sq
Model P1	$\overset{(0.03)}{PI = -1.56} + \overset{(0.0023)}{0.71\ \text{Source}} + \overset{(0.0002)}{0.80\ \text{Comp.ch}}$	0.68
Model P2	$\overset{(0.0)}{PI = -3.62} + \overset{(0.004)}{0.65\ \text{Source}} + \overset{(0.003)}{0.80\ \text{Ch.desc}}$	0.45
Model P3	$\overset{(0.001)}{PI = -1.34} + \overset{(0.0001)}{0.59\ \text{Ch.desc}} + \overset{(0.0016)}{0.50\ \text{Loc.add}} + \overset{(0.009)}{0.44\ \text{Comp.ch}}$	0.75
Model P4	$\overset{(0.002)}{PI = -2.95} + \overset{(0.003)}{0.65\ \text{Loc.add}} + \overset{(0.001)}{1.02\ \text{Loc.ch}}$	0.50

Table 6: Project specific regression equations

The correctness is shown to improve substantially (see Table 7), compared to the general model results whenever easy changes = [1-2] (except for project P2). The results are only presented for a minimal undecidable interval. However, the interval could be widened as shown in the previous section in order to get even better correctness in the decidable intervals.

PROJECT MODEL RESULTS		
Indecision	IR = 0%	
Classification	clas. [1-2] / [3-5]	clas. [1-3] / [4-5]
Model P1	[2.39 2.80] : 88%	[3.35 3.74] : 88%
Model P2	[2.31 2.52] : 74%	[3.42 3.61] : 93%
Model P3	[2.45 2.54] : 87%	[3.47 3.55] : 92%
Model P4	[2.45 2.54] : 81%	[3.47 3.55] : 89%

Table 7: Classification results

10006788L

5 Conclusions, Lessons Learned and Future Research

This modeling approach provides a simple and flexible way of classifying changes during the maintenance process. The classification power of continuous explanatory variables can be optimized by changing the class boundaries until the chi-square statistic reaches a maximum (this can be automated). This is performed while minimizing the number of metric classes and thereby facilitating the prediction process. It allows for an optimal use of the available explanatory variables by considering the uncertainty associated with each of them at the time of prediction.

A user defined loss function (i.e. risk model) can be minimized while selecting the decision boundaries on the predicted index range until a predefined expected loss is reached.
This allows the construction of a classification model optimal and customized, for specific user needs. Thus, by tuning the undecidable interval, he / she can handle in an appropriate and simple way the uncertainty associated with the model output. Also, the modeling process has shown many opportunities for a high extent of automation that would help optimize the metric class definitions and select the most suitable decision boundaries.

Despite the fact that collecting change effort data on a discrete range (i.e. ordinal level) makes the data analysis more difficult and the usable statistical techniques less powerful, valuable information can still be extracted from the data while taking into account the constraints associated with a software development environment. As presented, effective classification has been performed among three effort classes with respect to changes within the maintenance process.

Despite organizational issues and data collection accuracy problems, it would be better to collect effort data at a ratio level. This would allow the use of more effective statistical techniques. The gains in terms of management efficiency are likely to be substantial. However, if effort data are collected in a discrete manner, each class should contain, to the extent possible, the same number of changes. When the distribution is not uniform, classification for small proportion classes may be difficult.

Sub-system and component characteristics that are collectible in an automated way through code static analyzers (i.e. data binding between components, code complexity, ...) are likely to help refine the classification models. Maintainer skills and experience with respect to the maintained system should also be considered in the analysis in order to better select the required level experience for minimizing the cost of maintenance. Despite encouraging average results in the above experiments, a more complete data collection process is required in order to refine these change difficulty prediction models.

6 Acknowledgements

We would like to thank Jon Valett from the NASA Goddard Space Flight Center, Adam Porter and Chris Hetmanski for their suggestions that helped improve both the content and the form of this paper.

7 References

[BSP83] V. Basili, R. Selby and T. Phillips. "Metric Analysis and Data Validation across FORTRAN Projects". IEEE Transactions on Software Engineering, SE-9(6):652-663, November 1983

[BW84]V. Basili and D. Weiss. "A Methodology for Collecting Valid Software Engineering Data". IEEE Transactions on Software Engineering, SE-10(6):728-738, November 1984

[CAP88] J. Capon, "Statistics for the Social Sciences", Wadworth publishing company, 1988.

[DIL84] W. Dillon and M. Goldstein, "Multivariate Analysis", John Wiley & sons, 1984.

[GRA87] R. Grady, "Software Metrics: Establishing a Company-Wide Program", Prentice-hall, 1987.

[LS80] B. Lientz and E. Swanson, "Software maintenance management", Addison-Wesley, 1980.

[RUV92] D. Rombach, B. Ulery and J. Valett, "Toward Full Cycle Control: Adding Maintenance Measurement to the SEL", Journal of systems and software, May 1992.