The Lines of Code Metric as a Predictor of Program Faults: 
A Critical Analysis

Taghi M. Khoshgoftaar  
Department of Computer Science  
Florida Atlantic University  
khoshgo@servax.bitnet

John C. Munson  
Division of Computer Science  
University of West Florida  
jmunson@uwf.bitnet

Abstract

The relationship between measures of software complexity and programming errors is explored. Four distinct regression models were developed for an experimental set of data to create a predictive model from software complexity metrics to program errors. The Lines of Code metric, traditionally associated with programming errors in predictive models, was found to be less valuable as a criterion measure in these models than measures of software control complexity. A factor analytic technique used to construct a linear compound of Lines of Code with control metrics was found to yield models of superior predictive quality.

Introduction

In an attempt to characterize the internal nature of computer programs, many measures of software attributes have been developed and investigated. These software metrics represent quantitative descriptions of program attributes. Some of these metrics have been shown to be related, somehow, to quantitative measures of program reliability. Just as there are many software metrics, there are many different models of software reliability. Each of these models fundamentally study the computer program as a black box. The intrinsic assumption underlying each of the models is that the black box has one or more parameters that are uniquely determined for that particular box. Initial parameter estimates for these models do not reflect prior knowledge on the structure of the black box. There is now sufficient evidence to suggest that complexity metrics do provide substantial information on the distinguishing differences among the software systems whose reliability is being modeled and may be used in the reliability model development.

The foundations for the predictive model development based on software complexity metrics are clear. The model of computer software reliability will use the information available from software complexity metrics. There is clearly a relationship between measures of software complexity and errors during the program development and operational phases[2]. However, there are no viable models at present that reflect program complexity and potentials for program complexity in the prediction of error phenomena. There is a clear intuitive basis for believing that complex programs have more errors in them than simple programs. There is reasonable evidence to support the conclusion that computer software complexity metric models may be integrated with software reliability models.

Software complexity metrics have been shown to be closely related to the distribution of errors in program modules. That is, there is a direct relationship between some complexity metrics and the number of changes attributed to errors later found in test and validation [12]. Many researchers have sought to develop a predictive relationship between complexity metrics and errors. In particular, a close relationship has been found between the software measure of lines of code and errors. Previous studies [5, 15] have developed estimates of fault rates ranging from .3 to .5 faults per hundred lines of code. Lipow [10] developed a model which predicts the number of faults per line of code based on Halstead’s [7] software science metrics. Gaffney [6] proposed formulas relating the number of faults to the number of lines of code and to the number of conditional jumps. Crawford, et.al. [3] found that no one software complexity measure provided significantly better results than the LOC metric in predicting the number of faults in C software. They suggested that multiple variable models are necessary to find metrics which are important in addition to program size. Akiyama [1] related program faults to the decisions and calls (jumps) as opposed to the number of program statements in an assembly language environment. We are interested then, in investigating the relationship between the numbers of errors in programs and the particular software metrics which may be used to predict these errors.

In this study, we compare four different approaches to the problem of estimating errors in programs based on complexity metrics. Two of these approaches represent models based solely on the LOC metric. Two predictive models are based on complexity metrics. To reduce the dimensionality of the software complexity problem we have conducted an empirical review of the software complexity literature. Through the examination of many different programs, an underlying structure of the complexity problem space has emerged. Our previous work suggests the existence of possibly five basic components to program complexity [11]. Any program may be characterized by its associated measures on each of these underlying complexity domains. The second set of predictive models, then, shows the application of this new complexity view to the problem of predicting software errors. We have found that this approach yields models of superior predictive quality to those constructed with the LOC metric alone.

This paper will identify some of the basic problems associated with the development of predictive models for errors using complexity metrics. A basic statistical tool which will be used
to determine this relationship is that of regression analysis. As will be shown, there are many methods of developing a regression model for the same data. The choice of a particular model should be viewed as the selection of a best model from a pool of candidate models.

The basic regression model is based on the assumption that the independent variables of the analysis are not linear compounds of each other nor share an element of common variance. Two variables sharing a common element of variance are said to be collinear. To meet this assumption of non-multicollinearity, another statistical procedure called factor analysis may be used. The specific value of factor analysis is that the technique will reduce a data matrix to a set of orthogonal variables or factors that are, in fact, non-collinear.

When there is a complete absence of linear relationship among the independent variables, they are said to be orthogonal. In most regression applications the independent variables are not orthogonal. Usually the lack of orthogonality is not serious enough to affect the analysis. However, in software development the independent variables, software complexity metrics, are so strongly interrelated that the regression results are ambiguous. Typically, it is impossible to estimate the unique effects of individual software complexity metrics in the regression equation. The estimated values of the coefficients are very sensitive to slight changes in the data and to the addition or deletion of variables in the regression equation. The regression coefficients have large sampling errors which affect both inference and forecasting that is based on regression model. The condition of severe nonorthogonality is also referred to as the problem of collinearity in the software development. When confronted with a large number of variables measuring a single construct, it may be desirable to represent the set by some small number of variables that convey all or most of the information in the original set. The principal components are constructed so that they represent transformed scores on dimensions that are orthogonal.

A goal of software reliability modeling is to predict accurately the probable number of errors in computer programs. The software undergoes several stages of testing during program development phases. At each stage, corrections and modifications are made to the software with the hope of increasing its reliability. It is possible, however, that a particular modification, or a series of modifications, could lead to a deterioration in the reliability of the software due to increasing program complexity. The important statistical issue is how to model and describe changes in the reliability of the software as a result of the modifications. These reliability models, in turn, may be used in conjunction with cost information on program development. As a result, they may be used as very precise management tools.

The Relationship Between Program Complexity and Errors

The focus of this investigation is the nature of the relationship between measures of software complexity and measures of known errors in programs. The ability to model this relationship is desirable for a number of reasons. Given the viability of the predictive models and the fact that these measures may be obtained early in the software design process, program modules most likely to contain errors may be identified as they are prepared. To the extent that a definitive relationship between the software metrics and an assessment of known errors can be established, these metrics will serve as leading indicators of program reliability. Also, program modules that will later prove difficult to maintain may also be identified.

The Data Description

Data provided by B. Kitchenham [9] will be useful in the modeling of the relationship between the domain of complexity metrics and software errors. These data were collected on a set of 226 programs for a data communication software system. Each of metric values for these programs was collected by a manual inspection of the individual programs and their procedures. A description of the specific quantitative complexity data collected for each program is as follows:

- the number of programs which call a given program module (FI),
- the number of programs called by a specified program (CALL),
- the number of input parameters on the program interface (PI),
- the number of output parameters on the program interface (OP),
- the number of data structures the program reads from, but does not also write to (DR),
- the number of data structures the program writes to, but does not also read from (DW),
- the number of data structures the program both reads from and writes to (DB),
- program size in lines of code (non-comment, non-blank lines) in program (LOC),
- program control flow measured in terms of the number of branches (CF).

From these metrics, two additional synthetic metrics were computed. These were variations of the fan-in and the fan-out information flow metrics proposed by Henry and Kafura [8] defined as follows:

- the fan-in of a procedure is a tally of the number of procedures which may call the procedure, local flows into the procedure, plus the number of data structures from which this procedure receives information (MIFI),
- the fan-out of a procedure is a tally of the number of procedures called by the procedure, the data structures that the procedure updates, plus the number of output parameters (IFO).
Program errors may be divided into two classes; errors which have been found and fixed and unknown or unseen errors. The category of interest in this study is

- number of known errors, the number of faults corrected in the program, (KE).

The Predictive Models

Four distinct predictive models were developed and analyzed to study the relationship between errors and complexity metrics. Two of these models were based solely on the LOC metric in that the close relationship between lines of code and program errors has been known for some time. Two other models were developed incorporating additional measures of program complexity.

The first of these regression models, known subsequently as Model 1, is based on Gaffney's work as follows:

\[ y = b_0 + b_1 \cdot \text{LOC} \]

where the coefficients \(b_0\) and \(b_1\) represent the slope and intercept respectively and \(y\) is the dependent measure which is KE. This non-linear regression model was derived empirically. As an extension of this basic conceptual framework, the second model investigated, Model 2, was also a non-linear model in LOC. This model may be summarized as follows:

\[ y = b_0 + b_1 \cdot \text{LOC}^2 \]

where the parameters, \(b_0\), \(b_1\), and \(b_2\) are determined by least squares estimation and \(y\) is the dependent measure of program errors.

The third regression model, Model 3, was derived from a multiple linear regression based on the complete set of raw complexity metrics. The major problem, here, is the identification of the best of many possible models using this set or subsets of complexity metrics. The most intuitively obvious technique to use in the identification of the appropriate subset of complexity metrics is to perform all possible regression analyses (the combinatorial solution). Notwithstanding the fact that this can be a very labor intensive process for a regression model with a large number of metrics, there still remains the problem of the selection of the best of the models so produced. Clearly, some evaluation standard must be applied. Also, this particular technique lends itself well to the selection of an inappropriate model due to spurious random variation of the complexity metrics not related in any way to the systematic variation of the program errors.

The next alternative for the development of a regression model, is to use a stepwise regression procedure which involves the systematic incorporation of metrics in the regression model in an iterative manner. Within this class there are essentially three methods. First, there is the stepwise regression analysis. In this procedure, an initial model is formed by selecting the metric with the highest simple correlation with the program errors. In subsequent iterations new metrics are selected for inclusion based on their partial correlation with the metrics already in the regression equation. Metrics in this model may be removed from the regression equation when they no longer contribute significantly to the explained variance. There must be an a priori level of significance (in this case, \(p < 0.05\)) chosen for the inclusion or deletion of metrics from the model. The second stepwise procedure is forward inclusion. In the case of this procedure, a metric once entered in the regression equation, may not be removed. The third technique, backward elimination, forms a regression equation with all metrics and then systematically eliminates the metrics, one by one, which do not contribute significantly to the model.

Stepwise procedures for the selection of metrics are useful tools for the development of a predictive model only in the circumstances of non-collinearity. We recommend a different set of procedures in the presence of collinearity. Once the collinearities are identified, a set of new metrics, common factors, can then be formed by using factor analysis. These new measures of underlying orthogonal complexity domains will not, then, be collinear. Then stepwise procedures are used to select the factors which are important for the prediction of an enumeration of programming errors.

The last model, Model 4, is based on the reduced, orthogonal complexity domains through the use of factor analytic techniques. This reduced model shows the most promise to the study of the prediction of errors in software. With this procedure, the set of independent variables are mapped onto a smaller number of orthogonal dimensions through the use of factor analysis. A significant value in this process, is the fact that multicollinearity among the variables is first eliminated by the factor analysis. Also, the factor analysis serves to reduce the apparent dimensionality of the set of independent variables.

From a regression analysis of variance perspective, this also will have the net effect of reducing the degrees of freedom due to regression in that fewer total variables are presented as independent variables to the regression model.

The functional basis for the use of factor analysis is to describe, if possible, the covariance relationships among complexity metrics in terms of a few underlying, but understandable, random quantities called factors [cf. 4]. Basically, the factor model is motivated by the following argument. Suppose complexity metrics can be grouped by their correlations. That is, all metrics within a particular group are highly correlated among themselves but have relatively small correlations with metrics in a different group. It is conceivable that each group of metrics represents a single underlying construct, or complexity domain, that is responsible for the observed correlations.

The technique of factor analysis concerns itself with estimating the factor loadings for the mapping of the complexity metrics on the complexity domains. Once the factor loadings have been obtained, the major task is to make the best interpretation of the common factors. It is important to note that the burden of interpretation lies on the observer and is not intrinsic in the factor analysis. Usually, it is relatively simple to observe the relationships of complexity metrics grouped by their association with a common factor and attach a name to this set.

An example of the application of factor analysis to the field of software complexity metrics may be seen in the authors' recent study. The most important conclusion that can be drawn from
This investigation is that the domain of complexity measures does not appear to be unrestricted. There are many software complexity metrics in the literature, but there are relatively few dimensions in the complexity measure space. It would appear perfectly reasonable to characterize the complexity of a program with a simple function of a small number of variables that convey all or most of the information in the original set. The principal components are constructed so that they represent transformed scores on dimensions that are mutually orthogonal.

Through the use of factor analysis, it is possible to have a set of highly related variables, such as complexity metrics, be reduced to a relatively small number of complexity dimensions. When this mapping is accomplished by factor analysis, the transformed and reduced complexity dimensions are in fact orthogonal. This definitively solves the problem of multicollinearity in subsequent regression analysis.

This fourth regression model was based on factor scores obtained from the factor analysis of a subset of complexity metrics related to program size (LOC) and module coupling (CF, IFO, IP, and MIFI). The results of this factor analysis is shown in Table 1 below. Three distinct values emerge from this factor analysis. The first factor we have chosen to call Volume/Control in that it has associated with it those metrics which have been associated with either a volume or a control dimension from our previous investigations. The second and third factors are uniquely associated with the number of input parameters and the fan-in metric respectively. The actual factor pattern presented in this table represents the results of a varimax factor rotation, which is an orthogonal rotation of the original factor structure.

### Table 1

<table>
<thead>
<tr>
<th>Metric</th>
<th>Factor 1</th>
<th>Factor 2</th>
<th>Factor 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOC</td>
<td>.932</td>
<td>.183</td>
<td>.056</td>
</tr>
<tr>
<td>CF</td>
<td>.914</td>
<td>.144</td>
<td>.114</td>
</tr>
<tr>
<td>KE</td>
<td>.826</td>
<td>.042</td>
<td>.045</td>
</tr>
<tr>
<td>IFO</td>
<td>.791</td>
<td>.138</td>
<td>.055</td>
</tr>
<tr>
<td>IP</td>
<td>.145</td>
<td>.981</td>
<td>.077</td>
</tr>
<tr>
<td>MIFI</td>
<td>.056</td>
<td>.074</td>
<td>.994</td>
</tr>
</tbody>
</table>

### Table 2

<table>
<thead>
<tr>
<th>Model</th>
<th>Source</th>
<th>Deg. of Freedom</th>
<th>Sum of Squares</th>
<th>Mean Square</th>
<th>F Statistic</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Regression Error</td>
<td>1</td>
<td>220</td>
<td>545.87</td>
<td>232.37</td>
</tr>
<tr>
<td></td>
<td>Corr. Total</td>
<td>221</td>
<td>778.23</td>
<td>220</td>
<td>171.98</td>
</tr>
<tr>
<td></td>
<td>Regression Error</td>
<td>1</td>
<td>222</td>
<td>509.79</td>
<td>221</td>
</tr>
<tr>
<td></td>
<td>Corr. Total</td>
<td>223</td>
<td>1050.51</td>
<td>222</td>
<td>114.42</td>
</tr>
<tr>
<td>2</td>
<td>Regression Error</td>
<td>1</td>
<td>224</td>
<td>504.42</td>
<td>223</td>
</tr>
<tr>
<td></td>
<td>Corr. Total</td>
<td>225</td>
<td>1050.51</td>
<td>224</td>
<td>208.48</td>
</tr>
</tbody>
</table>
The actual regression equations are shown in Table 3 below for each of the four models. The plots of each regression line as against the data for that model are shown in Figures 1 through 5. Model 1 is intrinsically a non-linear regression model in which the parameters $b_0$, $b_1$, and $b_2$ are determined empirically through least squares estimation. The derived exponential term was determined to be $b_2 = 1.279$. For Model 2, a simple linear regression derived the slope and intercept for the transformed metric, $\text{LOC}^{4/3}$ as per the work of Gaffney.

The regression equations for Models 3 and 4 were derived from the best regression models from combinatorial regression, stepwise regression, and backwards elimination. In both cases, for Models 3 and 4, a single regression model emerged as best from all three selection techniques. In the case of Model 3, the reduced factor model, only Factor 1 was selected by all three techniques. This factor represents the single shared attributed dimension of the program control flow metric, CF; and the modified informational fan-out metric, IFO. For Model 4, two of these raw complexity metrics were selected, again, by all three selection procedures. These were the CF metric, and the IFO metric.

<table>
<thead>
<tr>
<th>Model</th>
<th>Parameter</th>
<th>Estimate</th>
<th>Std Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Intercept</td>
<td>0.81</td>
<td>0.002</td>
</tr>
<tr>
<td></td>
<td>LOC$^{4/3}$</td>
<td>0.002</td>
<td>0.13</td>
</tr>
<tr>
<td>2</td>
<td>Intercept</td>
<td>1.24</td>
<td>0.06</td>
</tr>
<tr>
<td></td>
<td>Factor 1</td>
<td>0.13</td>
<td>0.03</td>
</tr>
<tr>
<td>3</td>
<td>IFO</td>
<td>0.13</td>
<td>0.03</td>
</tr>
<tr>
<td>4</td>
<td>CF</td>
<td>0.06</td>
<td>0.01</td>
</tr>
</tbody>
</table>

The Evaluation of Regression Models

The net result of the previous discussion is that most regression studies will produce more than one possible model. An excellent discussion of the model evaluation process is available from Myers [13]. The objective, now, is to be able to evaluate the several models in terms of their predictive value. In our particular case, we are interested in predicting the number of potential errors in a program module based on its relative complexity. There are several statistical measures of the performance of a regression model. In general, there are two distinct classes of these evaluation criteria. The first of these classes contains statistics developed from the regression analysis of variance. The coefficient of determination, $R^2$ is an example of such a statistic. Another approach, the PRESS statistic, is based on residual analysis.

Traditionally, the $R^2$ statistic is used almost exclusively in empirical studies in software engineering. There are some distinct problems associated with the use of $R^2$. In that $R^2$ is the ratio of the regression sum of squares to the total sum of squares which is constant for all regression models, $R^2$ can only increase as independent variables are added to a regression equation, whether or not they will account for a significant amount of variance in the dependent variable. A variation on the $R^2$ statistics is the adjusted $R^2$ (or $\text{Adj} \ R^2$) which does attempt to correct for the number of variables in the regression equation (regression degrees of freedom, d.f.). The $R^2$ statistic does not assess the quality of future prediction, only the quality of fit on the sample data.

The PRESS statistic is based on a systematic examination of the residuals. A residual is the difference between an observed value of the dependent variable $y$ and the value $\hat{y}$ predicted by the model. A major problem with regression modeling and the prediction of errors in a program relates to the fact that some programs have disproportionately few or many errors in them. These outliers have a tendency to override a least square model. The PRESS statistic provides the opportunity to investigate models controlling for the effects of these outliers. This statistic is developed from PRESS residuals, $e_i = y_i - \hat{y}_i$, where $y_i$ is the value of the $i^{th}$ dependent variable and $\hat{y}_i$ is the predictive value from a regression equation formed with all observations, except the $i^{th}$. Thus, if there are $n$ sample points, $n$ separate regression equations will be formed, each with $n-1$ observations. The choice of a regression model is determined by selecting the one with the lowest value of the Average PRESS statistic [13].

With these selection criteria in mind, the four regression models may be compared. To this end the PRESS statistic and the $R^2$ are presented in Table 4 below. It can be seen that the model with the best fit as determined by the $R^2$ statistic is Model 4. This model was based on the selection of variables from a set of raw complexity values. As can be seen in Table 1, however, these are both measures of a single underlying complexity domain.

<table>
<thead>
<tr>
<th>Model</th>
<th>PRESS</th>
<th>Ave PRESS</th>
<th>$R^2$</th>
<th>$\text{Adj} \ R^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>556.93</td>
<td>2.51</td>
<td>.30</td>
<td>.30</td>
</tr>
<tr>
<td>2</td>
<td>592.70</td>
<td>2.66</td>
<td>.44</td>
<td>.44</td>
</tr>
<tr>
<td>3</td>
<td>522.71</td>
<td>2.35</td>
<td>.34</td>
<td>.34</td>
</tr>
<tr>
<td>4</td>
<td>555.86</td>
<td>2.49</td>
<td>.65</td>
<td>.65</td>
</tr>
</tbody>
</table>

From the standpoint of the predictive quality of a model, as measured by the PRESS statistic, the best of the four models is the factor model, Model 3. This factor model contains simply the single factor score for the Volume/Control complexity domain.

The most apparent conclusion which may now be drawn, is that the simple lines of code metric is not a particularly good predictor of program errors. Neither of the models which were based on the LOC metric performed as well as the models based on the more complete set of metrics. In the case of the regression with all possible raw complexity metrics, the LOC metric did not contribute significantly to the formulation of a final regression model. The regression model with the best
predictive quality was the factor analytic model which contained one factor which was composed of a linear compound of the LOC metric with measures of control flow and informational fan-out. Apparently far more important complexity considerations, than the mere size of a program, are those associated with the control complexity or the control complexity and its interaction with program length.

Summary
The specific focus of this study has been to investigate some aspects of the relationship between program complexity measures and program errors which are found during development. The specific statistical vehicle chosen to measure this relationship was regression analysis. Within the framework of regression analysis we have examined two separate means of exploring the connection between complexity and errors. First, the regression models were formed from the raw complexity metrics. Essentially, these models confirmed a known relationship between program lines of code and program errors. The second methodology involved the more general case of the regression of complexity measures and measures of errors. From this more global perspective, we believe there is a relationship between program errors and complexity domains of program structure and size (volume). Further, the strength of this relationship suggests that predictive models are indeed possible for the determination of program errors from these orthogonal complexity domains.

The basic technique we have developed in this study relates to some major problems we have observed in the effort to develop reliable and meaningful predictors for program modules in terms of the number of errors that these modules might contain. Software complexity metrics certainly would be useful in this regard in that they are numerical measures which may be obtained prior to the test and validation of a program. As has been shown, these metrics are quite interrelated. They are also, for the most part, highly correlated with measures of program errors. This high correlation by itself is an unreliable indicator of the predictive quality of models used in the study of software development errors. In fact, the large correlations are certainly indicators of multicollinearity which can only confound attempts at developing predictive models.

Through the judicious use of factor analysis, predictors may be mapped onto orthogonal domains. Our experience in this area indicates that there are relatively few such complexity dimensions in the existing set of complexity metrics. Measures developed from the reduced complexity metric space may then be used to develop predictive models of error of relatively great predictive quality. These multivariate models represent reasonable enhancements to models based solely on the LOC metric alone.

The subject of predictive quality and appropriateness of the models, has also failed to receive the attention that it deserves in many reliability models we have studied. In general, there are many statistics which aid in the determination of the predictive quality of regression models. We have examined several in this paper. Typically, research in this area is driven only based on the $R^2$ measure of predictive quality. As we have seen, this statistic can only increase as new predictors are incorporated in a regression model whether or not they contribute to the overall predictive capability of the model. The PRESS statistic, on the other hand, does provide a good measure of predictive quality.

Acknowledgement
The authors would like to express their sincere thanks to Drs. Barbara Kitchenham and Hugh Browton for providing the data set which was used in this study.

References