Practical Assessment of the Models for Identification of Defect-prone Classes in Object-Oriented Commercial Systems Using Design Metrics

Giancarlo Succi, Witold Pedrycz, Milorad Stefanovic, James Miller

Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB

Abstract

The goal of this paper is to investigate and assess the ability of explanatory models based on design metrics to describe and predict defect counts in an object-oriented software system. Specifically, we empirically evaluate the influence of design decisions to defect behavior of the classes in two products from the commercial software domain. Information provided by these models can help in resource allocation and serve as a base for assessment and future improvements.

We use innovative statistical methods to deal with the peculiarities of the software engineering data, such as non-normally distributed count data. To deal with overdispersed data and excess of zeroes in the dependent variable, we use negative binomial and zero-inflated negative binomial regression in addition to Poisson regression.

Furthermore, we form a framework for comparison of models’ descriptive and predictive ability. Predictive capability of the models to identify most critical classes in the system early in the software development process can help in allocation of resources and foster software quality improvement. In addition to the correlation coefficients, we use additional statistics to assess a models’ ability to explain high variability in the data and Pareto analysis to assess a models’ ability to identify the most critical classes in the system.

Results indicate that design aspects related to communication between classes and inheritance can be used as indicators of the most defect-prone classes, which require the majority of resources in development and testing phases. The zero-inflated negative binomial regression model, designed to explicitly model the occurrence of zero counts in the dataset, provides the best results for this purpose.

1 Introduction

Software quality is a key element in the success of any software organization. Assuring high quality in development of large software systems has become an increasingly complex, time-consuming activity. Consequently, it is essential to focus the available resources on the most critical parts of the system.

Client’s satisfaction is crucial for commercial software systems. Ensuring that the product satisfies functional and other quality requirements is essential in achieving this goal. Early indication of most critical software modules and possible bottlenecks is a valuable asset in efficient management of commercial software projects.

A widely used set of software metrics for object-oriented systems is the suite proposed by (Chidamber and Kemerer, 1994), referred to as CK metrics. Clearly, it is valuable to empirically validate the ability...
of these metrics to help in identification of the classes that cause most of the defects in the system, and consequently consume a great portion of the effort in the development and testing activities.

There has been a lot of work in this area, both theoretical and empirical. (Li and Henry, 1993) employed the CK metrics (without CBO) for predicting maintainability of two commercial software systems developed in object-oriented Classic-Ada. (Basili et al., 1996) organized an experiment in the university environment to evaluate the ability of CK metrics to be used as predictors of fault-prone classes. (Chidamber et al., 1998) explored relation between CK metrics and the size and the development time for three financial applications. (Briand and Wüst, 1999) also investigated the impact CK metrics have on the time spent in development. They used data from the music editor developed in the university settings. More recently, (Ronchetti and Succi, 2000) applied the subset of CK metrics extracted from the analysis documentation of two telecommunication projects to model the size of the system. An overview of the related work in this area with comparison of the most important aspects of the different studies is shown in Table 5. Our paper contributes to the existing research on this subject by focusing on a specific application domain using data from an industrial project in commercial application domain.

For the application analyzed in this study, all the modifications of the classes caused by defects in software operation were recorded throughout the development process. We assume the number of modifications referring to defects represents a good estimation of the defect-proneness of the class.

The metrics we collected are the software size, represented by the number of source lines of code (LOC), and the set of CK metrics, measuring different aspects of software design, such as complexity, coupling, cohesion, and inheritance. We use this set of product metrics to identify the characteristics of the object-oriented design with the highest impact on the defect-proneness of classes. All the metrics are collected from the source code of a stable product release.

In statistical analysis and modeling of the dependent variable in our study, i.e., the number of defects for the class, we use methods appropriate for the count data. In order to deal with typical problems in modeling software characteristics, such as too high variance of the dependent variable and underprediction of the zero values, we apply the Poisson regression model (PRM), the negative binomial regression model (NBRM), and the zero-inflated negative binomial regression model (ZTNBRM). These models are employed for estimation of the number of defects from the design metrics (class 3 model; Fenton and Pfleeger, 1996).

Beside the coefficient of correlation between the model estimations and the observed data, and the dispersion parameter (see Section 2.4), we use Pareto analysis to further characterize the performance of the models. This analysis represents an additional method for assessment and comparison of the models’ ability to identify the most critical classes in the system.

This paper is organized as follows: An overview of the applied metrics and statistical methods used in this study is provided in Section 2. Section 3 discusses the evaluation of the construction of models to explain the collected data. The reference products and the experimental data are presented in Section 4. In Section 5, the regression models are applied for estimation of the number of defects for classes based on the extracted metrics. The analysis of the results is presented in Section 6. A comparison with the related work performed in this area is shown in Section 7. Conclusions and directions for future research are presented Section 8.

2 Background

In this chapter we discuss the metrics collection process, methods used in the statistical analysis and modeling, including Pareto analysis for evaluation of the resulting models.
2.1 Object-oriented metrics

As mentioned, we use the relatively simple and well-understood CK metrics suite. This set of six metrics shows a good potential in forming a complete measurement framework in an object-oriented environment (Mendonça and Basili, 2000). The CK suite includes the Depth of Inheritance Tree (DIT) for a class corresponds to the maximum length from the root of the inheritance hierarchy to the node of the observed class. Another metrics related to the inheritance is NOC representing the number of immediate descendants of the class in the inheritance tree. CBO is defined as the number of other classes to which a class is coupled through method invocation or use of instance variables. RFC is the cardinality of the set of all internal methods and external methods directly invoked by them. We use Number Of Methods (NOM) measure as in (Ronchetti and Succi, 2000). The number of internal methods is extracted instead of forming a weighted sum of the methods based on complexity. (Basili et al., 1996) and (Chidamber et al., 1998) refer to this measure as a WMC, using unit weight for methods. The lack of cohesion in methods (LCOM) is defined as the number of pairs of non-cohesive methods minus the count of cohesive method pairs, based on common instance variables used by the methods in a class. Since the analyzed code is written in C++, source lines of code are counted using semicolons.

A number of alternative object-oriented measures have been proposed. Some of them account for deficiencies of the CK metrics (Li, 1998). The metric suite proposed by (Li, 1998) consists of the number of ancestor classes (NAC), number of local methods (NLM), class method complexity (CMC), number of descendant classes (NDC), coupling through abstract data type (CTA), and coupling through message passing (CTM). (Marchesi 1998) introduces metrics for object oriented analysis models in UML. (Nesi and Querci, 1998) propose a set of complexity and size metrics for effort evaluation and prediction, providing also a validation for some of them. (Reyes and Carver, 1998) define an object-oriented inter-application reuse measure. (Shih et al., 1998) propose a concepts of unit repeated inheritance and inheritance level technique for measuring the software complexity of an inheritance hierarchy. (Bansiya and Davis, 1999) introduce Average Method Complexity (AMC) and Class Design Entropy (CDE) that measure the complexity of a class using the information content. (Kamiya et al., 1999) propose revised set of CK metrics for software with reused components. (Miller et al., 1999) propose four new measures of hierarchy, inheritance, identity, polymorphism, and encapsulation in an object-oriented design. (Teologlou, 1999) describes the predictive object points for size and effort estimation.

The set of object-oriented design metrics and the source lines of code count used in this paper are collected from the source code using WebMetrics, a software metrics collection system (Succi et al., 1998). This tool supports collection of procedural and object-oriented set of software metrics for multiple programming languages.

2.2 Regression models for the count data

In this study we deal with a count dependent variable (number of defects for a class) ranging on an absolute scale.

Treating count variables as continuous, although being common practice (Fenton and Neil, 1999; Gray and MacDonell, 1997), also endorsed by (Briand et al., 1996), may result in inefficient and biased models (Long, 1997).

The most common distributions applied to the count data are based on the Poisson and multinomial distributions (Lloyd, 1999). The Poisson distribution is particularly suitable for counting events occurring over time. In the corresponding PRM, the Poisson distribution determines the probability of a count, where the mean of the distribution is a function of the independent variables. PRM has been used in software engineering for modeling the number of faults (Graves et al., 2000) and the effort expressed in hours (Briand and Wüst, 1999). PRM requires equidispersion, i.e., equality of the
conditional variance and the conditional mean of the dependent variable. When conditions for the PRM are not met, e.g., in case of high conditional variance of the dependent variable, the Negative Binomial (NB) distribution and the associated NBRM can be used (Lloyd, 1999; Briand and Wüst, 1999).

It is common in software metrics data that the number of zeros exceeds the prediction of both PRM and NBRM. Zero-inflated count models explicitly model the number of predicted zeros (Lambert, 1990).

The following subsections explain these models.

2.3 Poisson model

The Poisson process is a simple model for occurrence of random variables that assumes the probability of an arrival in a small interval determined by the independent variables is determined only by the size of the interval, not on the history of the process to that time (Papoulis, 1991). A Poisson distribution is the distribution of the numbers of events resulting from a Poisson process.

With respect to the occurrence of the defects in the classes and the set of predictors used in the model, we assume that the requirements of the Poisson process about the individual and independent occurrence of events are satisfied to a reasonable extent.

The Poisson distribution for a dependent variable \( y \), and a vector of \( n \) independent variables \( x=(x_1, \ldots, x_n) \) is given with:

\[
Pr(y \mid x) = \frac{e^{-\mu} \cdot \mu^y}{y!}
\]

where \( \mu \) is the mean of the dependent variable \( y \).

The Poisson distribution requires equidispersion of data, that is, the conditional mean and the conditional variance of the dependent variable should be equal (Briand and Wüst, 1999; Lloyd, 1999):

\[ E(y \mid x) = Var(y \mid x) \]

In practice, the conditional variance of the dependent variable in the model is often higher than its conditional mean, i.e., the dependent variable is overdispersed. The main cause of the overdispersion is failure of the Poisson distribution to account for heterogeneity in the data. Beside other factors such as wrong selection of the regression function or outlier data points, over-dispersion also significantly influences the goodness of fit of the overall model (Lloyd, 1999).

The PRM accounts for heterogeneity in the data based on the observed characteristics of items, i.e., based on the independent variables. To fully define the PRM, a regression function describing the underlying pattern, i.e., the mean of the data has to be defined in combination with the error distribution. The goal of statistical analysis is to find a simple regression function that successfully models the main behavior of the data.

The exponential regression function, corresponding to the multiplicative model for the means, is commonly used with the Poisson distribution (Long, 1997; Lloyd, 1999). The conditional mean is given by:

\[
\mu(y \mid x) = e^{\beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n} = e^{x \beta}
\]

where \( \beta \) is the vector of model parameters.

2.4 Negative binomial model

Empirical data are often over-dispersed, i.e., the value of the conditional variance is higher than the conditional mean of the dependent variable in the PRM. The main reason for this is the lack of complete control over experiments, attributing a great part of the variability in the observed data to unknown sources. This is also known as unexplained heterogeneity.
Overdispersion results in over-estimated statistical significance of the predictors in the model (Cameron and Trivedi, 1986). An extension of the PRM, the negative binomial regression model, allows the conditional variance of the dependent variable to exceed the conditional mean.

The NBRM can be derived from the Poisson distribution based on the unobserved heterogeneity by accounting for the combined effect of unobserved variables omitted from the original model (Gourieroux et al., 1984). In the NBRM, the mean $\mu$ is replaced with the random variable $\overline{\mu}$:

$$\overline{\mu} = e^{x^\top\beta + \varepsilon}$$

where $\varepsilon$ represents a random error uncorrelated with $x$.

The relationship between $\overline{\mu}$ and the original $\mu$ is:

$$\overline{\mu} = e^{x^\top\beta} = \mu e^\varepsilon = \mu \delta$$

With assumption that $E(\varepsilon) = 0$, the expected count after adding the new source of variation is the same as it was for the PRM, i.e., $E(\overline{\mu}) = \mu$.

In the NBRM, there is a distribution of $\overline{\mu}$'s rather than a single value for a given combination of independent variables. Consequently the probability distribution function for $\delta = e^\varepsilon$ must be specified in order to solve the probability for the dependent variable. The resulting distribution is a combination of the Poisson distribution and another two probability distributions.

$$Pr(y | x) = \int_0^\infty Pr(y | x, \delta) \cdot g(\delta) d\delta$$

Gamma distribution, with a relatively simple closed form of the resulting distribution (with parameter $\nu$), is commonly assumed for $\delta$ (Long, 1997):

$$g(\delta) = \frac{\nu^\nu}{\Gamma(\nu)} \delta^{\nu - 1} e^{-\nu \delta} \text{ for } \nu > 0$$

The resulting combined NB distribution is given with:

$$Pr(y_i | x_i) = \frac{\Gamma(y_i + \nu)}{\nu^\nu \Gamma(\nu)} \left(\frac{\nu}{\nu + \mu_i}\right)^\nu \left(\frac{\mu_i}{\nu + \mu_i}\right)^{y_i}$$

where $\nu$ is a positive estimated parameter, and $\Gamma$ stands for Euler gamma function:

$$\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt$$

For the NB distribution, the conditional mean of the dependent variable remains the same while the conditional variance of the dependent variable becomes:

$$Var(y_i | x_i) = \mu(1 + \nu^{-1} \mu_i)$$

The conditional variance of the dependent variable in the NB distribution is quadratic in the mean and, since $\mu$ and $\nu$ are positive, it exceeds the conditional mean of the Poisson distribution. The $\nu^{-1}$ is usually referred to as the dispersion parameter $\alpha$ since increasing $\alpha$ increases the conditional variance of $y$. Consequently, a low value of $\alpha$ represents a low level of over-dispersion.

The NB distribution corrects three main sources of poor fit that are often found when the Poisson distribution is used. First, the variance of the NB-distributed dependent variable exceeds the corresponding variance of the Poisson distribution for the given mean. Second, the increased variance in the NB results in substantially larger probabilities for small counts. With increased $\alpha$, the probability
of zero values in the NB is increased. For sufficiently big \( \alpha \), the conditional mode for all the values of the independent variables becomes equal to 0. Finally, the probabilities for larger counts are slightly larger in the NB distribution. Consequently, the resulting NBRM is the most commonly used model based on the combination of the Poisson distribution with other distributions (Long, 1997).

An alternative way to derive the NB distribution is based on the contagion process (Long, 1997). Contagion is defined as a process where analyzed items with a set of the independent variables initially have the same probability of certain event, but this probability changes as events occur over time. Consequently, contagion violates the assumption of independence in the Poisson distribution. Both the unobserved heterogeneity and contagion can result in the NB distribution of the dependent variable. The heterogeneity can thus be referred to as spurious contagion.

The NBRM model can be estimated by the maximum likelihood method, maximizing the likelihood equation:

\[
L(\theta | y, X) = \prod_{i=1}^{n} Pr(y_i | x_i)
\]

2.5 Zero-inflated model

The underprediction of zeroes in the PRM is partially resolved by the NBRM's increased conditional variance for the same conditional mean. On the other hand, zero-inflated models change the mean structure in order to explicitly model the excessive occurrence of zero counts (Long, 1997). These models also result in increased conditional variance of the dependent variable.

Zero-inflated models introduce the possibility that different processes generate zero counts and positive counts. They assume the population is consists of two different groups. An item belongs to one of the groups with probability \( \psi \) and to the other with the complementary probability \( 1-\psi \). This probability is determined from the characteristics of the item (Lambert, 1992). Items in the first group always have zero counts. Such items are different from those that have zero counts with a certain probability. These items belong to the second group together with the items with non-zero counts. The concept of two groups represents a discrete, unobserved heterogeneity since it is not known to which of the two groups an item with a zero count belongs.

In the group with items that are not always equal to zero, the resulting counts are governed by the PRM or the NBRM. For the NBRM, zeroes in this group occur with probability:

\[
Pr(y = 0 | x) = \left( \frac{\nu}{\nu + \mu} \right)^\mu
\]

where \( \mu = e^{\beta g} \).

The overall probability of zeroes is a combination of the probabilities of zeroes from each group multiplied by the probability of an item belonging to a particular group.

The resulting model has the following form:

\[
Pr(y = 0 | x) = \psi(x) + (1- \psi(x))Pr(y=0|x)
\]

\[
Pr(y = 0 | x) = (1- \psi(x))Pr_{P}(y|x)
\]

where \( Pr_{P}(y|x) \) is the probability given by the PRM or NBRM.

In the combined zero-inflated model, probability \( \Psi \) is determined by either a probit or logit model \( \Psi = F(\gamma y) \), where \( F \) is the normal or the logistic cumulative distribution function, respectively (Long, 1997). In this paper, we use the ZINBRM based on logit probability for zero counts.

The corresponding maximum likelihood method is available for zero-inflated models. The predictor vector \( z \) can (but does not have to) be the same as the vector \( x \) in the original PRM or NBRM. Clearly,
if \( z \) and \( x \) are equal, i.e., if the same predictors are used in both parts of the model, the resulting model has twice as many parameters as the corresponding NRBM.

3 Evaluating Software Engineering Data

There are no universal and general criteria for assessment of software engineering models. It is necessary to select one or more models that are most suitable for the goals in the analysis and for particular software projects (Littlewood, 1981).

Applicability, effectiveness, and predictive ability are used as criteria for assessment of models (Tian et al., 1995). Applicability evaluates the performance of a model over time and across the different datasets, e.g., for different projects. Effectiveness of the model quantifies its ability to be fitted to the actual observations. The ability of the model to predict future behavior in the software development process, based on the historical data, is usually referred to as predictive validity or predictive ability of the model (Fenton and Pfleeger, 1997).

Goodness of fit, based on the sum of square errors, is commonly used measure of effectiveness of the model (Yamada et al., 1983). Other aspects of effectiveness can be evaluated correlation coefficients and graphical methods, such as Pareto Analysis.

Capability and simplicity are also important factors for practical assessment of different models (Pressman, 1991). Model capability estimates usefulness of the information that the model provides for the software development process. Clearly, it is desirable to have a model with parameters that have some physical meaning, which can be well understood and interpreted. For a model to be really useful, it is essential that data collection process is time- and cost-effective. This characteristic of the model is referred to as the model simplicity. For example, the information about the modifications in the software is usually easily available to project managers from the company’s software configuration management system, and our analysis is based on such dataset.

Following the approach similar to Succi et al. (2000), our goal is both to create an accurate description of influence of decisions made at the design time to later defect behavior of classes to serve as a base for assessment and future improvements, and to identify most critical classes in the system early in the software evolution.

Goodness and likelihood of fit are used to measure descriptive ability of the models, while Pareto analysis is applied to assess the predictive ability of the models.

A graphical method called Alberg diagram can be used to compare performance of the different models in terms of the criticality prediction for the classes in the system (Ohlsson and Alberg, 1996). This diagram is based on the Pareto principle (also referred to as 80/20 rule; Ebert and Baisch, 1998). Pareto rule states that a small number of modules (i.e., classes in our analysis) causes the major portion of the problems and, consequently, consumes the most effort in the system. Alberg diagram is formed by placing modules in decreasing order with respect to the number of defects. The x-axis is the percentage of the total number of modules, while y-axes represents the cumulative number of defects discovered in the corresponding classes. By comparing the curves formed using the observed data and the results estimated by the model, the effectiveness of the model in identifying critical classes can be assessed. It is reported that models with lower correlation coefficients sometimes provide better prediction with respect to the criticality of the analyzed modules (Ohlsson and Alberg, 1996).

3.1.1 Methods for model estimation

Although ordinary least squares (OLS) is the most frequently used method of estimation for regression models, its application for fitting the parameters of the model is justified if error distribution is assumed
to be normal. On the other hand, the maximum likelihood (ML) method provides a general solution for fitting of the model parameters for non-normally distributed data, when the underlying distribution is known or assumed. The OLS and the ML estimates of model parameters are approximately the same for the linear model if error distribution is assumed to be normal. ML is also applicable for models, such as PRM, where the variance of the data is not constant (Tryfos, 1998).

ML method is designed to maximize the probability that the model represents the best fit to the empirical data. This estimator is consistent, i.e., the probability that this estimator differs from the true parameter by an arbitrary small amount tends toward zero as the sample size grows. The variance of the ML estimator is the smallest possible among consistent estimators. This feature is usually referred to as asymptotic efficiency of the ML estimator. Thus, we use ML estimators for the models in our analysis.

Fitting models to the empirical data has two aspects – explaining the mean tendency in the data, and variability of the data. Ability of the model to describe the main trend is captured using the likelihood score, and capability of the model with respect to the variability in the empirical data is measured through the overdispersion parameter \( \alpha \).

4 Discussion of the experimental data

This study is focused on two software projects from the commercial application domain. Both projects are developed in C++ programming language; project A over two and a half years, and project B over 6 years. All the developers, approximately 50 developers for project A and 11 developers for project B. All developers had similar experience and education levels (equivalent to BSc in Electrical and Computer Engineering or Computer Science) were involved in the development process. No particular pattern was observed in the assignment of developers to different problems or parts of the system, suggesting no selection bias.

Project A consists of 150 classes and project B of 144 classes for which the set of object-oriented design metrics was extracted and the number of defects and changed LOC was recorded. Total size of the projects A and B is 23 and 25 KLOC (thousands of lines of code), respectively.

Summary of the descriptive statistics for the extracted metrics is provided in Table 1.

<table>
<thead>
<tr>
<th></th>
<th>Project A</th>
<th>Project B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Min</td>
<td>Max</td>
</tr>
<tr>
<td>CBO</td>
<td>0</td>
<td>111</td>
</tr>
<tr>
<td>DIT</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>LCOM</td>
<td>20710</td>
<td>428.91</td>
</tr>
<tr>
<td>NOC</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>NOM</td>
<td>0</td>
<td>205</td>
</tr>
<tr>
<td>RFC</td>
<td>0</td>
<td>336</td>
</tr>
<tr>
<td>LOC</td>
<td>1</td>
<td>1674</td>
</tr>
<tr>
<td>Defects</td>
<td>0</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 1: Descriptive Statistics of the extracted CK metrics and LOC for project A

A high level of statistically significant correlation is present between some of the extracted metrics, in particular CBO, NOM, RFC and LCOM, as shown in Table 2.
<table>
<thead>
<tr>
<th></th>
<th>CBO</th>
<th>DIT</th>
<th>LCOM</th>
<th>LOC</th>
<th>NOC</th>
<th>NOM</th>
<th>RFC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project A</td>
<td>0.18</td>
<td>0.12</td>
<td>0.16</td>
<td>0.12</td>
<td>-0.12</td>
<td>0.14</td>
<td>0.17</td>
</tr>
<tr>
<td>Project B</td>
<td>0.20*</td>
<td>-0.05</td>
<td>0.22*</td>
<td>0.37*</td>
<td>-0.03</td>
<td>0.35*</td>
<td>0.34*</td>
</tr>
</tbody>
</table>

Table 2: Correlation coefficients between the extracted metrics and number of defects

Number of modifications directly related to software defects for a class is used as a proxy for the defect-proneness of the class. We build the models for this external measure as the dependent variable and metrics from the CK set as predictors.

For project A, there are in total 287 defects recorded, ranging from 0 to 17 defects per class. Total of 514 defects are recorded for project B, with up to 41 defects per class. Most of the classes in both projects have low number of defects. Histograms of the observed data in Figure 1 show that the normal distribution cannot be assumed for the dependent variable. Since we are dealing with count data, we use PRM, NBRM, and ZINBRM.

![Histograms for Project A and Project B](image)

Figure 1: Histogram of the number of defects

The Pareto rule (Ebert and Baisch, 1998) is applicable with respect to the number of defects. Less than 30% of the classes in project A correspond to 80% of the total number of defects, while in project B only 2% of the classes accounts for 80% of defects in the system. The presence of this principle that a relatively small portion of the system consumes most of the effort is closer examined using Alberg diagrams, which implements Pareto analysis, are discussed in Table 5 (Ohlsson and Alberg, 1996).

5 Extraction and analysis of the models for number of defects

For modeling the number of defects, we use three regression models based on the extracted set of design metrics. In these models, the exponent of the regression function is a linear combination of the metrics used as predictors.

The regression function in the PRM can be written in the log-linear form:

$$ln(\mu) = x\beta$$

This suggests that the PRM can be approximated by the linear regression model:

$$ln(\gamma) = x\beta + \epsilon$$
Since $y$ can also take zero value, it is necessary to add a positive constant $c$ to the dependent variable before taking the log (Long, 1997). Values of $c$ equal to 0.1 and 0.5 are typically used. In our analysis, we use $c = 0.5$. The resulting regression model is given with:

$$\ln(y+c) = x\beta+c$$

The parameters of this generalized linear model are estimated using the method of ordinal least squares (OLS). Although potentially biased, OLS estimations can be used for approximation of the statistical significance of parameters in the corresponding PRM (King, 1988).

We first apply the stepwise regression method with the log-linear form of the PRM. All the available metrics are allowed to enter this model. In the stepwise regression, independent variables are selected to enter the model based on the $p$-value. $p$-value is a measure of the statistical significance, representing the probability the outcome of the analysis is just a result of chance. The lower the $p$-value is, the higher is the statistical significance of the result. The independent variable with the smallest $p$-value is entered at each step of the regression, if that value is sufficiently small. Variables already in the model are removed if their $p$-value becomes sufficiently large. The method terminates when no more variables are eligible for inclusion or removal. In this analysis, we use 0.01 $p$-value as the entry criteria for a variable to enter the model, and 0.05 $p$-value to remove variable.

The two PRMs (PRM OLS 1 and PRM OLS 2) identified by the stepwise method for both projects are based on RFC and DIT as predictors. Although colinearity between some of the metrics is relatively high, the two metrics in the identified models (RFC and DIT) are not highly correlated (Table 2). In both models, parameters associated with RFC have positive, highly significant values.

<table>
<thead>
<tr>
<th>Project A</th>
<th>Project B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Intercept</td>
</tr>
<tr>
<td>PRM OLS 1</td>
<td>-1.63*</td>
</tr>
<tr>
<td>PRM OLS 2</td>
<td>-0.09</td>
</tr>
</tbody>
</table>

Table 3: Coefficients of the PRM where statistical significance is indicated by the presence of an asterisk.

It is possible to argue the any model selection mechanism is flawed. In fact, several papers in the statistical literature report a significant bias in the general application of stepwise regression. To safeguard against this the selection was repeated with other selection models, and where the numerical results change the basic semantic selection is relatively robust. Hence, we believe that the presented interpretation of the data is the most likely and hence will form the basic for the remainder of this paper. Further, as is illustrated in Table 6, this approach to model selection is a relatively common approach within much of the empirical software engineering literature.

The application of the ordinal least squares (OLS) method for fitting the parameters of the generalized linear regression model is justified if error distribution is assumed to be normal. In this case the OLS and the maximum likelihood (ML) estimates of $\theta$ are approximately the same for the linear model. On the other hand, the ML provides a very general solution for fitting of the model parameters when the underlying distribution is known or assumed. The ML estimator is consistent, i.e., the probability that this estimator differs from the true parameter by an arbitrary small amount tends toward zero as the sample size grows. The variance of the ML estimator is the smallest possible among consistent estimators. This feature is usually referred to as asymptotic efficiency of the ML estimator. Thus, we use ML estimators for the models in our analysis.

Log-likelihood function for the ML estimation in case of the Poisson distribution is given by:

$$l(\theta) = \sum_{i=1}^{k} (y_i \cdot \ln \mu_i(\theta) - \mu_i(\theta))$$
The vector of model parameters \( \theta = (\theta_1, ..., \theta_n) \) is determined by maximizing this sum over the available data points.

Using the results given with the PRM OLS 1 and PRM OLS 2, we use the same pair of metrics (RFC and DIT) as predictors for building the other models. Instead of the OLS, we use more general ML method for estimation of the PRM, NBRM, and ZINBRM. In the ZINBRM, the logit model is used for prediction of zeroes in combination with the NBRM.

The comparison of the resulting univariate (with extension 1) and bivariate (with extension 2) models with respect to the correlation coefficient \( r \), dispersion parameter \( \alpha \), and Relative Square Error (RSE) is provided in Table 4.

<table>
<thead>
<tr>
<th>Project A</th>
<th>( r )</th>
<th>( \alpha )</th>
<th>RSE</th>
<th>( r )</th>
<th>( \alpha )</th>
<th>RSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRM OLS</td>
<td>0.40</td>
<td>13.47</td>
<td>0.79</td>
<td>0.42</td>
<td>13.24</td>
<td>0.79</td>
</tr>
<tr>
<td>PRM ML</td>
<td>0.71</td>
<td>2.07</td>
<td>0.79</td>
<td>0.75</td>
<td>1.85</td>
<td>0.79</td>
</tr>
<tr>
<td>NBRM</td>
<td>0.69</td>
<td>0.51</td>
<td>0.79</td>
<td>0.67</td>
<td>0.42</td>
<td>0.80</td>
</tr>
<tr>
<td>ZINBRM</td>
<td>0.78</td>
<td>0.46</td>
<td>0.80</td>
<td>0.78</td>
<td>0.39</td>
<td>0.79</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Project B</th>
<th>( r )</th>
<th>( \alpha )</th>
<th>RSE</th>
<th>( r )</th>
<th>( \alpha )</th>
<th>RSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRM OLS</td>
<td>0.28</td>
<td>0.47</td>
<td>9.15</td>
<td>0.29</td>
<td>0.65</td>
<td>9.12</td>
</tr>
<tr>
<td>PRM ML</td>
<td>0.24</td>
<td>0.32</td>
<td>9.57</td>
<td>0.23</td>
<td>0.48</td>
<td>9.84</td>
</tr>
<tr>
<td>NBRM</td>
<td>0.26</td>
<td>0.31</td>
<td>11.86</td>
<td>0.20</td>
<td>0.18</td>
<td>19.38</td>
</tr>
<tr>
<td>ZINBRM</td>
<td>0.23</td>
<td>0.24</td>
<td>11.82</td>
<td>0.21</td>
<td>0.16</td>
<td>19.24</td>
</tr>
</tbody>
</table>

**Table 4: Resulting models**

### 6 Analysis of the results and discussion

Two reasonably good PRMs based on RFC and DIT (univariate and bivariate) are identified by the stepwise regression. These models in the log-linear form are given with:

\[
\ln(\text{NumberOfDefects} + 0.5) = \beta_0 + \beta_1 \text{RFC} + \beta_2 \text{DIT}
\]

Where \( \beta_i \) parameters are zero for metrics that do not enter a specific model.

Corresponding NBRM and ZINBRM are built using the ML for estimation of parameters. Results in Table 4 suggest significant differences between the models and some quantitative improvements in compared to the baseline PRM OLS models.

First, a high level of overdispersion is present in the PRM OLS models. This result is not surprising for the software metrics data. However, overdispersion is significantly lower with ML estimators. The other result of the ML method is increased correlation between the model prediction and the observed data.

Models based on the NB distribution show even higher capability of dealing with overdispersion. This improvement is partially the result of increased probability of low and high values of the dependent variable with the NB distribution.

Although the NBRM deals with the overdispersion more successfully than PRM, the disadvantage is a lower resulting correlation with the observed data.

ZINB regression models result in the highest correlation coefficient and the lowest dispersion parameter. These models incorporate the capability to successfully predict zero values with the ability of the NB distribution to account for overdispersion. Improvements achieved by the ZINBRM come with the expense of doubling the number of model parameters compared to the original NBRM.
The resulting bivariate models in Table 4 show slightly better performance than univariate models. The additional variable enables bivariate models to better fit the observed data. However, this improvement is modest and depends on many factors, such as the underlying dataset.

![Histograms of RFC and DIT](image)

**Figure 2: Histograms of RFC and DIT**

Since the metrics are non-negative, positive values of model parameters suggest positive influence of the predictors to the dependent variable. The link between the dependent variable and predictors is exponential in the analyzed regression models.

All resulting models suggest high influence of communication between classes measured by RFC to the dependent variable in the analysis. The additional predictor, the measure of inheritance DIT, does not significantly increase the performance of the bivariate models compared to the univariate RFC-based models. In addition, model parameters associated with DIT have much lower statistical significance than RFC-related parameters.

Relatively narrow range of DIT also limits the influence of this measure to the resulting models. RFC is a measure with a wider range of values. In our case RFC ranged from 5 up to 250. Clearly, the higher the communication of a class with other classes, the higher will be the need for modifications in that class.

As mentioned Pareto analysis is used to compare the ability of different models for criticality prediction of the modules in the software system (Ohlsson and Alberg, 1996). Table 5 shows the Pareto analysis for the models employed on our dataset with respect to the number of defects. Pareto analysis on the observed number of defects shows that about 30% of the classes cause 80% of the total number
of the defects in the system. The parteo analysis shows that both univariate and bivariate models have relatively good performance in prediction of the defect-prone classes. Using any of those models, less than 50% of the classes can be identified that cause 80% of the defects in the system.

<table>
<thead>
<tr>
<th></th>
<th>Project A</th>
<th>Project B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Univariate</td>
<td>Bivariate</td>
</tr>
<tr>
<td>PRM OLS</td>
<td>45</td>
<td>43</td>
</tr>
<tr>
<td>PRM ML</td>
<td>45</td>
<td>43</td>
</tr>
<tr>
<td>NBRM</td>
<td>45</td>
<td>44</td>
</tr>
<tr>
<td>ZINBRM</td>
<td>45</td>
<td>44</td>
</tr>
</tbody>
</table>

Table 5: The Pareto analysis for the models and predicted percentiles of the classes with 80% of the defects in the system

As mentioned, RFC is a measure of communication between classes defined as the number of internal and external methods that can be executed in response to a message received by an object of the class. Clearly, a large number of methods, potentially invoked from a class, increases the complexity of the class and requires greater level of understanding and effort in development and debugging of the class.

Depth of inheritance tree represents the position the class has in the inheritance hierarchy, i.e. the maximum distance from the root node when multiple inheritance is supported, i.e. in C++. Although the deeper position of the class in the hierarchy increases its potential for reuse through inheritance, at the same time the increased number of inherited methods makes it more difficult to predict the behavior of the class. It can be beneficial to keep the level of inheritance relatively low, at expense of compromising the potential of reuse through inheritance, in order to enable easier testing and understanding of design and implementation.

While some of the object-oriented design metrics tend to change through the evolution of the project from the design phase to implementation, DIT is one of the measures determined by the architecture of the system and it typically stays unchanged in this process. Consequently, the depth of inheritance tree information can be used relatively early in the software development process.

In this paper, we have explored various different statistical approaches to modeling this problem. One obvious question is does any one of the approaches seem obviously superior to be others:

- Effectiveness of the model can be expressed in terms of correlation of its estimations with the observed data and in terms of RSE. PRM OLS and PRM ML models show best performance with respect to these criteria on both datasets.
- Part of the capability of the models to explain the empirical data is attributed to their ability to account for overdispersion and occurrence of zero counts. With respect to these criteria, using dispersion parameter $\alpha$ as a measure, ZINB and NBRM perform best on the two datasets.
- Pareto analysis can be used to assess both effectiveness of the models and their predictive ability. Pareto analysis is applied to data in this analysis demonstrate only minor differences in performance of the models. The analysis suggests that predictive ability of bivariate univariate models are similar in this respect.
Clearly, the comparison of any multi-faceted decision is highly complex and when we consider the competing measures of fitness, we can only see that no obvious “winner” emerges. Hence, we can only conclude, that this survey is inconclusive in this respect.

7 Related studies

This section provides a table with an overview of the related work in this area, comparing the most important aspects of different studies.

<table>
<thead>
<tr>
<th>7.1.1.1 Study</th>
<th>Li and Henri, 1993</th>
<th>Basili et al., 1996</th>
<th>Chidamber et al., 1998</th>
<th>Briand and Wüst, 1999</th>
<th>Ronchetti and Succi, 2000</th>
<th>This work</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment</td>
<td>Industrial</td>
<td>University</td>
<td>Industrial</td>
<td>University</td>
<td>Industrial</td>
<td>Industrial</td>
</tr>
<tr>
<td>Lifecycle phase of the independent variable</td>
<td>Design &amp; code</td>
<td>Code</td>
<td>Design &amp; code</td>
<td>Design &amp; code</td>
<td>Analysis</td>
<td>Design &amp; code</td>
</tr>
<tr>
<td>Programming language</td>
<td>Classic-Ada</td>
<td>C++</td>
<td>C++</td>
<td>C++</td>
<td>C++</td>
<td>C++</td>
</tr>
<tr>
<td>Application domain</td>
<td>User interface and scientific</td>
<td>Information system</td>
<td>Financial application</td>
<td>Music editor</td>
<td>Telecomm.</td>
<td>Commercial application</td>
</tr>
<tr>
<td>Size</td>
<td>2 projects</td>
<td>8 projects</td>
<td>3 projects</td>
<td>1 project</td>
<td>2 projects</td>
<td>2 projects</td>
</tr>
<tr>
<td>Dependent variables</td>
<td>Number of lines changed during maintenance as a proxy for maintenance effort</td>
<td>Probability of fault</td>
<td>Productivity, rework effort, design effort</td>
<td>Development time as a proxy for effort</td>
<td>Size as proxy for development effort</td>
<td>Number of defects for classes</td>
</tr>
<tr>
<td>Statistical analysis</td>
<td>Parametric linear regression</td>
<td>Logistic regression</td>
<td>Stepwise linear regression including dummy variables</td>
<td>Poisson regression and regression trees</td>
<td>Parametric and non-parametric correlation: linear regression</td>
<td>Poisson, negative binomial, and zero-inflated regression</td>
</tr>
<tr>
<td>Conclusions</td>
<td>No other CK metrics but CBO influence significantly the dependent variable</td>
<td>NOM, DIT, CBO, RFC significantly influence the dependent variable</td>
<td>High values of CBO and LCOM significantly influence the dependent variable</td>
<td>Size measures can be used as good predictors of effort</td>
<td>NOM significantly influences the dependent variable</td>
<td>RFC and DIT have highest impact on both dependent variables</td>
</tr>
<tr>
<td>Other considerations</td>
<td>N/A</td>
<td>The cross-correlation of the CK metrics are low</td>
<td>DIT and NOC assume low values; CBO, NOM, and RFC are highly correlated; the other correlations between metrics are low</td>
<td>Coupling measures do not add substantial quality to the models</td>
<td>DIT and NOC assume low values; the cross-correlation between the three CK metrics are low</td>
<td>Some cross-correlation between the CK metrics are high (NOM, CBO, RFC, LCOM)</td>
</tr>
</tbody>
</table>

Table 5: Related work
After all of this work and endeavour, we would hope that a clear picture has emerged about the usefulness of the CK metrics. Unfortunately, no clear unifying conclusion exists. An interesting further study, would be to undertake a multi-analysis of these studies to attempt to reveal any underlying numerical patterns consistent within the studies. It is also interesting to note, that the underlying statistical approach in not constant across the studies, and one strong possibility is that the individual findings are strongly influenced by the analytical approach. Further, more complex analytical approaches are possible, such as the use of hierarchical linear models, and one possibility that exists is that work within this field will have to more to these more involved approaches in at attempt to discover the true underlying nature of the observed data.

8 Conclusion

In this paper we provided an empirical validation of the common object-oriented design metrics used for early identification of the defect-prone and effort-intensive classes. Using the number of modifications caused by defects as an external measure, we investigated the impact that the object-oriented design has on the number of defects for a class. The design of an object-oriented software system offers a substantial amount of information about the system even before any coding has started. We provide the results of this extensive study carried out on a project from the commercial application domain.

In this analysis we use statistical models for the count data, i.e., PRM, NBRM, and ZINBRM. These models account for the typical problems with the software metrics data, such as overdispersion and heterogeneity. The ability of the models to identify the most critical classes is assessed using Pareto analysis.

Results indicate that coupling (RFC) and inheritance (DIT and NOC) represent best indicators of the defect-intensive classes. By using these two measures, project managers can identify the classes and files that require more careful development and rigorous testing. The trade-offs in software design can be better understood and managed by having precise quantitative insights into relationship between the conflicting aspects (Chidamber and Kemerer, 1994). Corresponding models can be used to improve both the development and testing process.

References:


Cameron A.C. and Trivedi P.K. (1986) “Econometric models based on count data: Comparisons and applications of some estimators and tests,” Journal of Applied Econometrics, 1, 29-93


Lambert D. (1990) "Zero-inflated poisson regression with an application to defects in manufacturing," Technometrics, 34, 1-14


Reyes L. and Carver D. (1998) 'Predicting object reuse using metrics', SEKE '98, Tenth International Conference on Software Engineering and Knowledge Engineering, Skokie, IL, USA

