A study on fault-proneness detection of Object-Oriented systems

F. Fioravanti, P. Nesi
Department of Systems and Informatics, University of Florence
Via di S. Marta 3, 50139, Florence, Italy
fioravan@dsi.unifi.it, nesi@ingfi1.ing.unifi.it

Abstract

Fault proneness detection in object-oriented systems is an interesting area for software companies and researchers. Several hundreds of metrics have been defined with the aim of measuring the different aspects of object-oriented systems. Only a few of them have been validated for fault detection, several interesting works with this view have been considered. This paper reports a research study started from the analysis of more than 200 different object-oriented metrics extracted from the literature with the aim of identifying suitable models for the detection of fault-proneness of classes. Such a large number of metrics allows extracting a subset of them in order to obtain models that can be adopted for fault proneness detection. To this end, the whole set of metrics has been classified on the basis of the measured aspect in order to reduce their number to a manageable one; then statistical techniques have been employed to produce a hybrid model comprised of 12 metrics. The work has been focussed on identifying models that can detect as many faulty classes as possible and, at the same time, models that are based on a manageable small set of metrics. A compromise between these aspects and the classification correctness of faulty and non-faulty classes was the main challenge of the research. As a result, two models for fault-proneness classes detection have been obtained and validated.

Index terms: *object-oriented metrics, maintenance, fault estimation, empirical validation.*

1 Introduction

Since the adoption of the Object-Oriented paradigm, several metrics and suites have been defined and validated in order to cover the different aspects of system development e.g., [1], [3], [5], [6], [7], [13], [17], [19], [22], [24]. These metrics have been defined and validated for different goals: effort estimation and prediction, reusability, maintenance, etc.

Some studies of metrics and measurement frameworks for object-oriented systems have been presented in [6], [10], [12], [17], [18], [19], [24], where general concepts for the estimation of system size, complexity and reuse level have been proposed together with many other metrics.

Fault-proneness detection is an interesting area. Quality and maintenance effort control depend on the understanding of this concept. In the last years, a lot of effort and work have been performed in order to define suitable metrics and models for fault detection [1], [3], [4].

This paper reports a research study of more than 200 different object-oriented metrics extracted from the literature with the aim of identifying suitable models for fault-proneness classes detection. The considered object-oriented metrics have been estimated by using three assessment tools for the analysis of C++ code. These measuring tools are: TAC++, developed at the Department of Systems and Informatics of the University of Florence [6], [10], [19]; CPP-Analyzer, developed by Harry M. Sneed [21]; and M-System, [1], [3], [4] developed at FHG-IESE. These tools estimate the metrics defined by the corresponding research teams but also several other metrics published in the literature, for a total of 226 different metrics at class level.

In this paper, an investigation on how all these metrics are related to fault-proneness detection and how many of these metrics are needed to obtain a good model for fault detection has been performed. The work has been focussed on identifying models that can predict as many errors as possible and at the same time are comprised of a manageable small set of metrics. To this end, the finding of a satisfactory compromise between these aspects and the classification correctness of faulty and non-faulty classes was the main challenge. The research study has been based on the estimated metric values and on a set of projects developed at the University of Maryland.

To this end, two models for fault-proneness detection based on logistic regression have been defined. These have

0-7695-1028-0/01 $10.00 © 2001 IEEE

been validated against a well-known set of projects in the literature, and compared with similar models already defined in the literature [4]. This work can be regarded as an extension and a confirmation of the results obtained in [4]. One of the results of this paper is that only few of the 226 metrics are relevant for obtaining a good identification of faulty classes in small-medium sized projects. To this end, the metrics have been classified on the basis of the measured aspects to reduce their number to a manageable one. This operation has reduced the number of metrics to be considered to 42, obtaining a first model for fault-proneness detection. This first model has been considered unsuitable since the number of metrics was still too high to be effectively usable. Starting from this model, statistical techniques based on logistic regression have been employed to produce a hybrid model comprised of only 12 metrics.

The large number of metrics selected as independent variables and the several aspects taken into account by them suggest that a hybrid model with metrics that considers different aspects can be suitably employed. This statement has been also partially supported in [4] where the hybrid model obtained a score non completely satisfactory with respect to other models. This paper confirms that a model (hybrid-model) that covers several distinct aspects related to the object-oriented paradigm (coupling and cohesion, structural, functional) can be suitably applied to faulty class identification obtaining a very high value of confidence. The experiments reported in this paper and in [4] and [5], were based on the same data set such as briefly described in the following. The main novelties of this paper are: the adoption of a different process to identify the model and the usage of PCA for reducing the number of metrics to a manageable one.

This paper is organized as follows. In Section 2, the study and the data adopted are presented. In Section 3, the statistical techniques adopted during the study are discussed. In Section 4, the proposed models are presented together with a discussion and comparison with other models presented in the literature. Conclusions are drawn in Section 5.

2 Empirical Study Background

In this section, the variables involved in the empirical study are presented together with the main goals of the study reported in this paper.

2.1 Dependent Variable

The systems adopted for the validation of the models defined during this study were developed by students participating in the upper division undergraduate/graduate level course at the University of Maryland. These projects and data were used several other times for studying quality models for object-oriented systems [1], [4], [5]. The students were grouped in 8 groups, and they were asked to develop a medium-sized management information system in C++ on the basis of a waterfall life cycle model and applying OMT [20] as the Object-Oriented Analysis/Design method (several libraries were used during the development). Details can be recovered in [1]. The testing phase was performed by an independent group of experienced software professionals that provided feedback about the detected faults by means of filled forms. This set of projects has been identified as a test case for the reasons reported in the following. It is quite difficult to recover large object-oriented industrial projects with a detailed list of faults detected during and after the development. The definition of new models has to deal with the comparison of the models already presented in the literature, for that reasons we have choosen to adopt this data-set.

The data related to faults have been used as the dependent variable for this study. In particular, a class has been considered faulty if at least a fault has been detected. Since the number of cases against which the model could be fitted was limited, no further inspection on the relationships between faulty classes with 2 or more faults and metrics have been carried out.

The number of cases collected in the data set was of 113 classes, with about one third of faulty classes.

2.2 Independent Variables

The independent variables are several object-oriented metrics calculated by means of the three tools for the analysis of C++ code mentioned in the introduction.

The cited tools are capable of processing C++ sources for producing measures of 226 metrics: 82 extracted by TAC++, 68 by CPP-Analyzer and 76 by M-System. All these metrics could be potentially considered as independent variables for the definition of models for estimation and prediction of fault-proneness in object-oriented systems. The considered metrics cover all the aspects of object-oriented measures presented in the cited literature. For lack of space, it is impossible to describe all these metrics and cite all the technical papers in which they are defined and discussed.

At more general level, CPP-Analyzer is mainly focused on the evaluation of functional and high-level quality metrics such as object points, and several other metrics related to Quality and Functional code estimation.

M-System evaluates metrics mainly related to class cohesion and coupling estimation and covers also the structural and design aspects relevant for object oriented systems. The considered metrics are mainly extracted from suites presented in the literature: Chidamber and Kemerer

[7], Li and Henry [17], Lee et al. [23], Bieman and Kang [2], and several others.

The TAC++ tool generates complexity, size, structural and cognitive metrics for effort estimation and prediction, completing the covering of the possible metrics suitably employed for the assessment of object-oriented systems: the suite of metrics proposed by the authors in the past [6], [10], [11]; some Henderson-Sellers metrics [12], [13]; some Thomas and Jacobson metrics [22], etc.

2.3 Target of the Study

The main goal of this study has been to define and validate models for fault-proneness detection. The starting point has been to collect the measures of different sets of object-oriented metrics available in the literature (as discussed above). These metrics have been used to identify the most suitable model for fault-proneness detection by using statistical techniques and tools, such as Principal Component Analysis and Logistic Regression.

In particular, the planned goals can be summarized as follows:

- to define a model with the aim of obtaining a suitable accuracy in the fault proneness detection (the target was a detection capability greater than 90%) disregarding the number of metrics involved in the model. This was defined in order to see if that value can be reached or not, even considering a very high number of metrics.

- to refine and modify the previous model in order to obtain the lowest degradation possible in performance (at least a detection capability of 80%) and reducing the number of metrics employed in the model to a manageable number.

The first model can be adopted to verify which metrics are useful for fault-proneness detection and which metrics cover all aspects related to fault identification, while the second model can be applicable in real conditions for the number of metrics involved. The models can be used to highlight which metrics and aspects are more relevant for the faulty problems of object-oriented systems.

3 Statistical Techniques for Data Analysis

In order to obtain a suitable model for fault proneness detection two commonly used statistical techniques have been adopted. The former is the Principal Component Analysis, while the latter is the multivariate logistic regression. In the following subsections, these techniques are briefly discussed together with their principal contributions and drawbacks.

3.1 Principal Component Analysis

In the analysis of a data set with many variables it can be difficult to reduce the number of variables that explain the variability of the data under analysis without losing information. This result can be performed by using Principal Component Analysis (PCA) [9]. Principal Components (PCs) are the orthogonal linear combinations of variables whose variance is equal to those observed. The first PC is the linear combination of variables which explain the maximum amount of variance in the data set. The other PCs are orthogonal to the previously extracted components and explain in turn the maximum amount of the residual variance.

Usually only a small subset of all the variables has a large coefficient (loading) in a PC, and therefore only these variables should be considered significant from a statistical point of view. The variables having a high loading usually identify the dimension captured, even if a certain degree of interpretation is required. In order to simplify the structure of the PCs, a rotation of the components is usually performed. This operation is focused on the reduction of the loading of the coefficients that were small in the component matrix and increases the loading of the already significant components.

In this paper, the *Varimax* rotation method has been adopted [16]. This rotation process works on the component columns allowing a reduction of the number of significant variables for each component. The criteria represented in Fig.1 and described in the following have been adopted for extracting the PCs from the data set under analysis.

- *Identification of variables/metrics to be included in the analysis.* Usually all variables/metrics are included in the analysis, except some specific variables that can be excluded *a-priori* because they are not statistically significant (i.e., they have a null variance).

- *Definition of the matrix on which to perform the analysis.* Usually the correlation matrix is adopted, because it is symmetric and contains the correlation of all variables.

- *Definition of the number of factors to be extracted.* The maximum number of factors is the rank of the correlation matrix, even if in general a lower number of factors is typically considered.

- *Definition of criteria for stopping the extraction of factors.* The commonly used procedure consists in extracting the PCs since the eigenvalue reaches a predefined value, or stopping their extraction when a sufficient variance is explained.

- *Application of the rotation criteria.* The rotation methods can be orthogonal or oblique and they are targeted

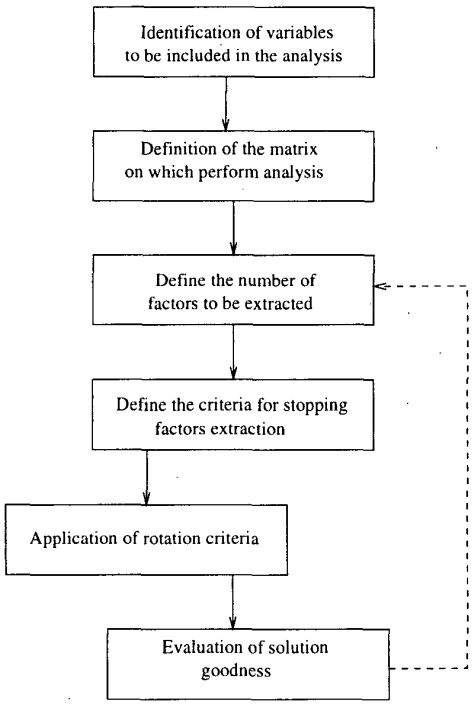

Figure 1. Sequence of steps to perform a factorial analysis by using PCs.

to simplify the factors' interpretation depending on the target chosen (i.e., *Varimax* method is useful for reducing the number of variables for each component, while *Quartimax* method is more focused on enhancing the contribution of the most significant variable among all the PCs).

- *Evaluation of solution goodness.* Statistical indexes and/or graphical representations can be adopted to judge the suitability of extracted PCs for the problem under analysis.

In order to apply the above mentioned steps, it is necessary to define a criterion to select a certain number of PCs that explain a good percentage of the variance. Several criteria can be suitably employed. The most commonly used are: (i) to fix the number of PCs in advance, (ii) to fix the target explained variance, (iii) to fix the eigenvalue related to a component. In this study, the second approach has been adopted and the cut-off value for the total explained variance eigenvalue has been imposed at 85%.

3.2 Multivariate Logistic Regression

Multivariate logistic regression is a standard technique to estimate the regression function that better matches the probability of owning a dichotomic (or binomial) attribute by a set of independent variables [15]. The result of the regression is always a number between 0 and 1, and the obtained value can be mapped to the target binomial attribute (0 or 1) by using a cut-off. The logistic regression function is represented in equation (1):

$$logit(\pi(x)) = \beta_0 + \sum_{i}^{q} \beta_i x_i, \tag{1}$$

where β_i are the coefficients evaluated by the logistic regression process (in particular β_0 is the constant to be included in the model), x_i are the metrics values and $logit(\pi(x))$ is the ratio between the success and fault probabilities:

$$logit(\pi(x)) = \ln\left[\frac{\pi(x)}{1-\pi(x)}\right], \tag{2}$$

and $\pi(x)$ is the probability that the attribute assumes the value 1 as a function of the set of independent variables. Please note the $logit(\pi(x))$ ranges from $-\infty$ to $+\infty$, while the related probability is always comprised between 0 and 1 since the probability of the target attribute can be described like a logistic function:

$$\pi(x) = \frac{e^{X\beta}}{1+e^{X\beta}}. \tag{3}$$

In Fig.2, the shape of the function used to represent the sigmoid of *logit* is depicted. This shape is particularly useful because it tends to the limit values (0 and 1) gradually.

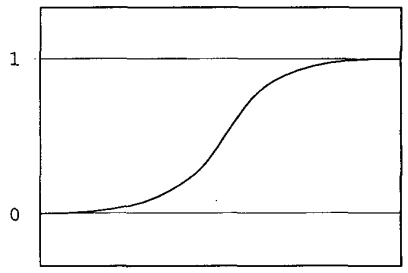

Figure 2. Sigmoid of *logit* function.

It should be noted that the analysis of the results of a dichotomic variable is based on the evaluation of Tab.1, where the classification performed by means of the chosen model is represented against the observed or expected value.

Several statistical coefficients (figures) can be evaluated on the basis of the previous Tab.1. In particular, the most relevant figures are:

	Dependent Variable (Y)			
Target Attribute (X)		0	1	
	0	a	b	a+b
	1	c	d	c+d
		a+c	b+d	n

Table 1. Table for the study of the results about the significance of a dichotomic variable. In the following 1 means *Fault*, and 0 means *No-Fault*. "a" and "d" represent the correct estimations while "b", "c" are the false-positives and false-negatives, respectively.

- *Bravais-Pearson* correlation coefficient:

$$\phi = \frac{ad - bc}{\sqrt{(a+b)(c+d)(a+c)(b+d)}} \quad (4)$$

that measures the symmetry between Y and X;

- *Sensibility*, that is the ratio of the fault-prone classes correctly classified:

$$Sen = \frac{a}{a+c} \quad (5)$$

- *Correctness*, that is the ratio of non fault-prone classes correctly classified:

$$Cor = \frac{d}{b+d} \quad (6)$$

- *Completeness*, that is the ratio of fault-prone classes and the total number of fault-prone classes:

$$Com = \frac{d}{c+d} \quad (7)$$

- *Classification Correctness*, that is the percentage of correctly classified items:

$$Cla = \frac{a+d}{n} \quad (8)$$

- *False Negative Percentage*, that is the percentage of classes predicted as non-faulty, while they are faulty:

$$fnp = \frac{c}{n} \quad (9)$$

- *False positive Percentage*, that is the percentage of classes predicted as faulty, while they are non-faulty:

$$fpp = \frac{b}{n} \quad (10)$$

All these figures can be calculated in order to evaluate the result goodness of regression models. In order to increase some of them it is necessary to adjust the cut-off value that is used for discriminating between 0 and 1 values (no faulty and faulty value in our case). The logistic regression model tends to fit better the largest group. An appropriate adjustment of the cut-off value equilibrates this trend.

4 Results Analysis

In this section, a short description of the procedures adopted for the data analysis and for the definition of the models is reported together with the analysis of the results obtained with the figures selected for the evaluation of the goodness of proposed models.

4.1 Correctness Classification Model

As a first step, all metrics available from the above mentioned tools have been considered. In this phase, no limitation on the number of metrics was employed. For this reason, the 226 metrics calculated by the three adopted tools have been analyzed with descriptive statistic techniques in order to identify totally meaningless variables in the set. On the basis of descriptive statistic, all the variables with a null variance have been removed. The eliminated metrics were 5 (1 from TAC++ and 4 from CPP-Analyzer); these metrics measured quantities that were not present in the analyzed code, such as the number of *goto*, or the number of *unions* or the number of protected attributes defined as pointers.

A number of 221 metrics is too huge to be realistically applied in real industrial cases.

The metrics have been clustered in groups on the basis of the aspects that they mainly address. The classification was performed considering the following groups: cohesion and coupling, structural, functional, cognitive, object-oriented size and complexity. The last group includes metrics that are based on classical size or complexity metrics and takes also into account structural aspects of the class definition, see for examples [11], [19]. This approach is typical of size and complexity metrics for object-oriented systems.

The classification of the original 226 metrics into these groups has produced the distribution reported in Tab.2.

Measured aspects	num. of metrics
Functional	45
Structural	49
Coupling and Cohesion	52
Object-oriented Size and Complexity	45
Cognitive	35

Table 2. Metrics grouping on the basis of the measured aspects.

After the removal of the 5 non-significant metrics previously identified, the PCA has been applied to each group of metrics extracting components until a variance not lower than 85% was obtained. The estimation of the PCA has produced reliable results since the number of variables, v, and that of measures, m, satisfied the rule $m/v > 2$. This rule is particularly restrictive but assures a suitable level of result reliability.

The most relevant metrics of each PC have been selected for the inclusion in an integrated group of metrics addressing the several aspects considered. This process has brought to the elimination of several marginally influent metrics with respect to the application of the PCA to each group, collecting a final number of 68 metrics, classified according to Tab. 3

Measured aspects	num. of metrics
Functional	15
Structural	14
Coupling and Cohesion	17
Object-oriented Size and Complexity	12
Cognitive	10

Table 3. Metrics clustering after the PCA on each group.

Several models have been built by grouping the 68 metrics in different combination of 50 metrics each, taking out 18 metrics every time. For each restricted model the logistic analysis has been performed in order to verify its significance as a model for fault proneness detection.

At the end of this stochastic approach, a significant model with 50 metrics has been identified. From this model of 50 metrics, other 8 metrics have been removed by obtaining a model of 42 metrics by using the PCA. This has only marginally decreased the capability of the model in terms of fault proneness detection. In Tab.4, the result of the model with 42 metrics is reported. Please note that the cut-off value has been imposed to 0.5 and no improvement was obtained by changing that value.

	Predicted			
Actual	No-Fault	Fault		Prediction %
No-Fault	59	1	60	98.33%
Fault	2	51	53	96.23%
	61	52	113	
Overall				**97.35%**

Table 4. Model with 42 metrics, cut-off to 0.5. The reported numbers are related to faulty classes and not to the number of faults identified.

Note that the first goal was reached since a percentage of correct classification greater than 90% has been obtained. Only 3 misclassifications were obtained in this case. In Tab.5, the figures for evaluating the goodness of the logistic regression model are reported. All the parameters reported in Tab.5 are very close to suitable values usually adopted for the validation of a logistic regression model, confirming the great accuracy and validity of the proposed model.

The metrics of the above described model can be classified according to several criteria: the tool to which the metrics belong, the aspect they cover, the type of metric, etc. In Tab.6 a classification of the 42 metrics involved in the model is reported. The name of the metrics is not significant at this stage, while they are detailed and commented in the second model proposed.

figure	value	optimal value
ϕ	0.947	1
Sen	0.967	1
Cor	0.981	1
Com	0.962	1
Cla	0.974	1
fnp	0.017	0
fpp	0.009	0

Table 5. Figures obtained for the model with 42 metrics.

Tool	number of metrics
$TAC++$	10
$M - System$	18
$CPP - Analyzer$	14

Metric Type	numb. of metrics
Functional	13
Structural	9
Coupling and Cohesion	13
Object-oriented Size and Complexity	5
Cognitive	2

Table 6. Classification of model with 42 metrics on the basis of the aspects covered by the metrics.

4.2 The Reduced Model

The model presented in Section 4.1 has a great identification capability, but its usage in real conditions for system assessment is computationally difficult for the high number of metrics involved.

The target of the reduced model is to employ a lower number of metrics without substantially decrease the predictive capability under the target value of 80%. Several techniques can be adopted to reduce the number of independent variables of a model: (i) application of PCA, (ii) evaluation of correlation matrix and (iii) adoption of stepwise logistic regression.

The second method can be questionable since there is no evidence that two correlated metrics do not have to belong to the same model. Therefore, after considering that a selection of metrics based on PCA and classification have been already performed, the third method has been applied.

Stepwise logistic regression can be backward or forward. In our case the backward elimination based on the Wald test has been adopted since this method allows eliminating from the models the less significant variables from a statistical

point of view. This process is quite different from that used in [4]

The application has allowed to substantially reduce the number of metrics. The obtained model is comprised of 12 metrics and the results are reported in Tab. 7, where a cut-off of 0.37 has been adopted. Please note the classification percentage is closer to 85%.

	Prediction			
Actual	No-Fault	Fault		Prediction %
No-Fault	47	13	60	78.33%
Fault	4	49	53	92.45%
	51	62	113	
Overall				**84.96%**

Table 7. Reduced model with 12 metrics and cut-off to 0.37. The reported numbers are related to faulty classes and not to the number of faults identified.

Also in this case, the planned goals have been correctly reached since a percentage of correct classification greater than 80% with no more than 10 metrics has been obtained. In Tab.8, the figures for evaluating the goodness of the logistic regression model are reported.

figure	value	optimal value
ϕ	0.785	1
Sen	0.921	1
Cor	0.790	1
Com	0.924	1
Cla	0.849	1
fnp	0.035	0
fpp	0.115	0

Table 8. Figures obtained for the model with 12 metrics.

Also in this case the figures reported in Tab.8 are close to the optimal values confirming that also a model with a lower number of metrics can be suitably employed for fault detection and prediction.

The metrics involved in this model are reported in the following list:

- CO (M-System), Connectivity [14] that is a metric defined in terms of graph elements built upon LCOM metrics;
- ICP_L (M-System), Information-flow/based Coupling [17] that is the number of called class local methods weighed by the number of parameters of the methods.
- $LCOM1$ (M-System), Lack of Cohesion in Methods 1 [7] that is the number of couples of methods that do not use attributes in common;
- $LCOM2$ (M-System), Lack of Cohesion in Methods 2 [7] that is the number of methods couples that do not use attributes in common minus the number of couples that do it;
- NAI (TAC++), Number of Attributes Inherited;
- $NAML$(TAC++), Number of Attributes and Methods Locally defined;
- $NDSTT$ (CPP-Analyzer), Number of different Statements Types [21]: it counts the number of different types declared;
- NFR (CPP-Analyzer), Number of function references [21];
- $NMImp$ (M-System), Number of Methods Implemented (local, inherited, overridden);
- RFC_∞: (M-System), Response set For Class [7]: that is the set of methods belonging to a class plus the set of methods directly or indirectly invoked by class methods.
- $STMTS$ (M-System), Statements that is the number of declarations in the class methods;
- TCC (M-System), Tight Class Cohesion [2]: this metric measures the number of attributes "indirectly" used by a method, that is all the attributes that are used by methods invoked by the method.

This model contains metrics belonging to all the above-mentioned tools for metric estimation and covers several distinct aspects of object-oriented metrics: (i) coupling and cohesion (CO, ICP_L, $LCOM1$, $LCOM2$, NFR, RFC_∞ and TCC), (ii) structural aspects (NAI, $NAML$, $NMImp$), (iii) functional aspects related to method code ($NDSTT$, $STMTS$).

Please note that all the variables reported in Tab.9 should be considered as significant since the greatest $p - value$ is lower than 0.10. This fact strengths the model that can be considered quite reliable from the statistical point of view.

The model proposed is hybrid, since it presents metrics producing values with different measurement units, such as several others presented in the literature. The correct unit type is obtained by using weights, β, of the logistic regression formula, that also play the role of converting factors. In Tab.9, the weights to be applied for obtaining the classification in the logistic regression model are reported together with their standard errors and Wald test with significance. Please note that these weights are the βs of equation (3).

In Tab.10, the univariate logistic regression for the same 12 metrics is reported together with the statistical figures. The univariate analysis is useful in order to verify the predictive capability of each metric involved in the model as

Metric	Weight	Stand.Error	Wald-test	$p-value$
CO	-7.897	2.672	8.733	0.003
ICP_L	0.051	0.017	8.862	0.003
$LCOM1$	0.078	0.031	6.121	0.013
$LCOM2$	-0.75	0.029	6.618	0.010
NAI	0.866	0.342	6.404	0.011
$NAML$	-0.721	0.310	5.406	0.020
$NDSTT$	-0.108	0.033	10.399	0.001
NFR	0.066	0.032	4.359	0.037
$NMImp$	0.503	0.292	2.961	0.085
RFC_∞	0.102	0.037	7.670	0,006
$STMTS$	0.022	0.009	6.496	0.011
TCC	3.943	2.180	3.271	0.070

Table 9. Coefficients for the multivariate logistic regression model with statistical figures. The $Log-Likelihood$ value is -46.116 and R^2 is 0.432.

Metric	Weight	Stand.Error	Wald-test	$p-value$
CO	-0.846	0.689	1.505	0.220
ICP_L	0.024	0.079	9.705	0.002
$LCOM1$	0.022	0.017	1.612	0.204
$LCOM2$	0.001	0.017	0.353	0.552
NAI	0.058	0.044	1.717	0.190
$NAML$	0.036	0.019	3.437	0.064
$NDSTT$	0.027	0.008	11.878	0.001
NFR	0.038	0.012	10.147	0.001
$NMImp$	0.053	0.026	3.987	0.046
RFC_∞	0.070	0.020	12.400	0,000
$STMTS$	0.015	0.004	16.284	0.000
TCC	-0.316	0.547	0.334	0.563

Table 10. Coefficients and statistical figures for the univariate logistic regression, for the 12 metrics adopted in the reduced model.

a unique predictor. Some of the variables have a strong predictive capability even when they are singularly taken to define a model, such as in the univariate analysis. For example, $STMTS$ and RFC_∞ present a correctness close to 70%.

It should be noted that the $p-values$ obtained in the multivariate case are always significant, while often in the case of univariate logistic regression the significance is not good enough.

Similarly to what has been evidenced in the models proposed in [4], some coefficients change sign in the univariate with respect to the multivariate case. This is due to suppressor relationships among the variables, which are commonly observed in multivariate regression analysis [8]. For the same reasons non-significant variables in the univariate model could result significant in the multivariate model. The differences in the sign for metrics that present non significant relevance in the univariate analysis have no sense.

In Tab.11 the residuals for the misclassifications are reported. It should be noted that the residuals are high in absolute value confirming that an adjustment of the cut-off does not increase the prediction capability of the model.

Actual	Pred. Value	Residual
No-Fault	0,4091	-0,4091
No-Fault	0,7876	-0,7876
No-Fault	0,8927	-0,8927
Fault	0,0614	0,9386
No-Fault	0,3703	-0,3703
No-Fault	0,6852	-0,6852
No-Fault	0,6509	-0,6509
No-Fault	0,7568	-0,7568
No-Fault	0,9135	-0,9135
No-Fault	0,5064	-0,5064
Fault	0,2011	0,7989
No-Fault	0,6089	-0,6089
Fault	0,2444	0,7556
No-Fault	0,5822	-0,5822
Fault	0,1822	0,8178
No-Fault	0,6182	-0,6182
No-Fault	0,4694	-0,4694

Table 11. Residual evaluation for misclassifications

4.3 Comparison with Other Models

In [4], three models for fault-proneness detection have been presented. $Model\ I$ is based on size metrics, $Model\ II$ is based on coupling, cohesion and inheritance metrics, while $Model\ III$ is a "mixed", hybrid model. In [4] a significant work was presented by using the same data, a reduced number of metrics and a slightly different approach to reduce their number. The approach adopted by Briand *et al.* [4] for obtaining the models can be summarized in the following steps:

1. Optional selection of the metrics on the basis of the aspects/features covered – such as: cohesion, coupling, inheritance, etc.;
2. Selection of the most promising metrics on the basis of the $p-values$ of Univariate Logistic Regression;
3. Multivariate Logistic Regression with the metrics obtained by the previous selections.

The approach followed for obtaining the models presented in this paper has been quite different. The starting point for the model definition was the idea to use a hybrid model. Hybrid models adoption seems to be the best approach for taking into account all aspects. To this end, we started from 226 metrics, with the aim of covering as many aspects as possible. Even with the idea of highlighting the relevant aspects with fault detection:

- Selection of the metrics divided according to the measured aspects;

- Principal Component Analysis for identifying metrics having the stronger loading within the same aspect;
- Identification of a hybrid model by considering the best representative metrics for each aspect of the classification considered.
- Multivariate Logistic regression with backward technique for eliminating the lower significant metrics on the basis of the obtained $p - values$.

The different approach followed in this paper with respect to that of Briand et. al. deals with the adoption of the PCA for the selection of the most promising metrics instead of using univariate logistic regression. The assumption made in [4] is that only the metrics that have a strong statistical significance in the univariate logistic regression should be included in a multivariate model. As can be seen by the comparison between Tabb.9 and 10, this assumption is not always true since metrics (i.e. LCOM2, TCC) with a poor relevance in the univariate inspection demonstrate their validity if inserted in a more complex model.

The approach followed in this paper to select metrics by the means of PCA in order to choose metrics covering orthogonal aspects has allowed to insert in the final models also metrics that have been discarded by the Briand methodology.

In Tab.12 and 13, the comparison among the models proposed in this paper and those of [4] is reported according to the figures already adopted in this paper. In the table, Model-42 and Model-12 refer to the above-presented models with 42 and 12 metrics, respectively. Please note the differences of values between the figures obtained for the proposed model and the models available in the literature.

	ϕ	Sen	Cor	Com	Cla	fnp	fpp
Model-42	0.947	0.967	0.981	0.962	0.974	0.017	0.009
Model-12	0.785	0.921	0.790	0.924	0.849	0.035	0.115
Model I	0.350	0.650	0.604	0.604	0.628	0.186	0.186
Model II	0.676	0.833	0.811	0.811	0.823	0.085	0.085
Model III	0.633	0.814	0.778	0.792	0.796	0.097	0.106

Table 12. Comparison in terms of statistical figures among different models.

At a general level, the obtained results confirm what other researchers have obtained and are comparable in terms of numerical results. The models presented in this paper are better ranked with respect to those of the other discussed models, confirming the validity of the approach adopted to obtain the proposed models. Although the difference in percentage is not so strong, a significant result is that a suitable model has to include a set of metrics covering coupling, cohesion and structure aspects, while complexity, size and functional aspects can be less relevant.

Model	number of metrics	% Total	% Correctly Classified	
			No-Fault	Fault
Model-42	42	97.35	98.33	96.23
Model-12	12	84.96	78.33	92.45
Model I	3	62.83	65.00	60.37
Model II	7	82.30	83.33	81.13
Model III	9	80.53	81.67	79.24

Table 13. Comparison among the prediction percentage of the models under comparison.

5 Conclusions

A new approach to define models for fault-proneness detection and prediction has been presented together with the statistical techniques adopted for their definition. This paper reports a research study on more than 200 different object-oriented metrics extracted from the literature. On the basis of this large set of metrics extracted by three different tools for the analysis of C++ code, two new metric models for fault-proneness detection have been presented and validated against well-known test cases in the literature.

The first model reaches a great accuracy (more then 97% of correctly predicted classes) in terms of classified items even if the number of involved metrics is relevant (42 metrics). The second model involves only 12 metrics with a classification correctness close to 85% in terms of faulty classes. The metrics involved in this second model have been presented in this paper together with the corresponding weights to be applied for obtaining the logistic regression equation.

The comparison with other similar models presented in the literature confirms the relevance of the proposed models in terms of accuracy of the results. Another significant result is that a hybrid model is suitable and has to include coupling, cohesion and structure metrics in order to be suitably employed, while the aspects related to complexity, size and functional aspects can be less relevant. Further studies in this field are in progress, in order to refine and extend the presented models to better identify the metrics that address the fault-proneness detection and create and validate models for non object-oriented programming.

Acknowledgments

The authors would like to thank: Prof. V. Basili and Prof. L. Briand for providing data related to test projects and comments, Dr. H. M. Sneed for providing his CPP-Analyzer tool adopted during the development and validation of the proposed models, and FGH-IESE for providing us the M-System tool. Thanks also to colleagues that provided several comments of an early version of this work. A warm thanks to the people who have worked in the past for the implementation and test of the several components of our code analyzer, TAC++.

This work was partially supported by Ex60% Italian Ministry of University and Scientific Research.

References

[1] V. R. Basili, L. Briand, and W. L. Melo. A validation of object oriented design metrics as quality indicators. *IEEE Transactions on Software Engineering*, pp.751-761, Oct. 1996.

[2] J. Bieman and B. Kang. Cohesion and reuse in an object-oriented system. In *Proc. ACM Symp. Software Reusability (SSR'94)*, pp.289-262, 1994.

[3] L. C. Briand, J. W. Daly, and J. K. Wust. A unified framework for coupling measurement in object oriented systems. *IEEE Transactions on Software Engineering*, vol.25, n.1, pp.91-120, January 1999.

[4] L. C. Briand, J. Wust, J. W. Daly, and D. V. Porter. Exploring the relationships between design measures and software quality in object oriented systems. *Journal of Systems and Software*, 1998.

[5] F. Brito e Abreu and W. Melo. Evaluating the impact of object oriented design on software quality. In *Proc. of the 3rd IEEE International Software Metrics Symposium*, Berlin, Germany, March 1996.

[6] G. Bucci, F. Fioravanti, P. Nesi, and S. Perlini. Metrics and tool for system assessment. In *Proc. of the IEEE International Conference on Complex Computer Systems*, pages 36–46, California, USA, August 1998.

[7] S. R. Chidamber and C. F. Kemerer. Towards a metrics suite for object-oriented design. In *Proc. of OOPSLA'91, 6th Conference on Object-Oriented Programming Systems, Languages, and Applications, ACM SIGPLAN NOTICES VOL. 26, N.11*, Phoenix, USA, October 1991.

[8] R. Darlington. Multiple Regression in Psychological Research and Practice. In *Psychological Bulletin*, vol.69, n.3, pp.161-182, 1968.

[9] G. Dunteman, Principal Component Analysis, Sage University Paper, 07-69, Thousand Oaks, CA, USA, 1989.

[10] F. Fioravanti, P. Nesi, and S. Perlini. Assessment of system evolution through characterization. In *Proc. of the IEEE International Conference on Software Engineering*, pp.456-459, Kyoto, Japan, April 1998.

[11] F. Fioravanti, P. Nesi, and F. Stortoni. Metrics for Controlling Effort During Adaptive Maintenance of Object Oriented Systems, In Proc. of the IEEE International Conference on Software Maintenance, IEEE Press, Oxford, England, pp.483-492, Sept. 1999.

[12] B. Henderson-Sellers. Some metrics for object-oriented software engineering. In *Proc. of the International Conference on Technology of Object-Oriented Languages and Systems, TOOLS 6 Pacific 1991*, pp.131-139. TOOLS USA, 1991.

[13] B. Henderson-Sellers, D. Tegarden, and D. Monarchi. Metrics and project management support for an object-oriented software development. In *Tutorial Notes TM2, TOOLS Europe'94, International Conference on Technology of Object-Oriented Languages and Systems*, Versailles, France, 7-10 March 1994.

[14] M. Hitz, B. Montanzeri, Measuring coupling and Cohesion in Object-Oriented Systems, In *Proc. Of the International Symposium on Applied Corporate Computing*, Monterrey, Mexico. http://www.pri.univie.ac.at/ hitz/papers/ESEC95.ps

[15] D.W. Hosmer, S. Lemeshow. Applied Logistic Regression. John Wiley & Sons, 1989.

[16] H. F. Kaiser. The varimax criterion for analytic rotation in factor analysis. *Psychometrika*, vol.32, pp.187-198, 1958.

[17] W. Li and S. Henry. Object-oriented metrics that predict maintainability. *The Journal of Systems Software*, vol.23, pp.111-122, 1993.

[18] J.C. Munson, T.M. Khoshgoftaar. The Detection of Fault-phone Programs. *IEEE Transactions on Software Engineering*, vol.18, n.5, pp.423-432, 1992.

[19] P. Nesi and T. Querci. Effort estimation and prediction of object-oriented systems. *The Journal of Systems and Software*, Vol.42, pp.89-102, 1998.

[20] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Object-Oriented Modeling and Design*. Prentice Hall International, Englewood Cliffs, New Jersey, 1991.

[21] H. M. Sneed. Estimating the costs of software maintenance tool. In *Proc. International Conference on Software Maintenance*, PP.168-181, Opio, France, October 17-20 1995.

[22] D. Thomas and I. Jacobson. Managing object-oriented software engineering. In *Tutorial Note, TOOLS'89, International Conference on Technology of Object-Oriented Languages and Systems*, pp.52, Paris, France, 13-15 Nov. 1989.

[23] S. W. Y. Lee, B. Liang and F. Wang. Measuring the coupling and cohesion of an object oriented program based on information flow. In *Proc. International Conference on Software Quality, Maribor, Slovenia*, 1995.

[24] H. Zuse. *A Framework of Software Measurement*. Walter de Cruyter, Berlin, New-York, 1998.