JOURNAL OF OBJECT TECHNOLOGY

Online at http://www.jot.fm. Published by ETH Zurich, Chair of Software Engineering ©**JOT, 2007**

Vol. 6, No. 10, November-December 2007

Investigating effect of Design Metrics on Fault Proneness in Object-Oriented Systems

K.K.Aggarwal, Yogesh Singh, Arvinder Kaur, Ruchika Malhotra
University School of Information Technology, Guru Gobind Singh Indraprastha University, Kashmere Gate, Delhi 110006, India

Abstract

Demand for quality software has undergone with rapid growth during the last few years. This is leading to an increase in the development of metrics for measuring the properties of software such as coupling, cohesion or inheritance that can be used in early quality assessments. Quality models that explore the relationship between these properties and quality attributes such as fault proneness, maintainability, effort or productivity are needed to use these metrics effectively. The goal of this work is to empirically explore the relationship between object-oriented design metrics and fault proneness of object-oriented system classes. The study used data collected from Java applications is containing 136 classes. We use a set of twenty-six design metrics in our work. Result of this study shows that many metrics are based on comparable ideas and provide redundant information. It is shown that by using a subset of metrics in the prediction models can be built to identify the faulty classes. The proposed model predicts faulty classes with more than 80% accuracy.
Keywords: Measurement, Metrics, Object-Oriented, Coupling, Cohesion, Inheritance, Empirical Analysis.

1 INTRODUCTION

There are several metrics proposed in the literature for capturing the quality of Object-Oriented (OO) design and code, for example, ([Aggarwal05]; [Braind98][Braind99]; [Bieman95]; [Cartwright00]; [Chidamber94][Chidamber91]; [Harrison98]; [Henderson96]; [Hitz00]; [Lake94]; [Li93]; [Lee95]; [Lorenz94]; [Tegarden95]). These metrics provide ways to evaluate the quality of software and their use in earlier phases of software development can help organizations in assessing large software development quickly, at a low cost [Braind99]. But how do we know which metrics are useful in capturing important quality attributes such as fault-proneness, effort, productivity or amount of maintenance modifications. Empirical studies of real systems can provide relevant answers. There have been few empirical studies evaluating the effect of object-oriented metrics on software quality and constructing models that utilize them in predicting quality attributes in the system, such as (Basili96] [Binkley98]; [Braind00][Braind01]; [Cartwright00]; [Chidamber98]; [Emam99][Emam01]; [Gyimothy05]; [Harrison98]; [Li93]; [Ping02]).

More data based by empirical studies, which are capable of being verified by observation or experiment are needed. The evidence gathered through these empirical studies is today

Cite this article as follows: K.K Aggarwal, Yogesh Singh, Arvinder Kaur, Ruchika Malhotra, "Investigating effect of Design Metrics on Fault Proneness in Object-Oriented Systems", in *Journal of Object Technology*, vol. 6, no. 10, November-December 2007, pp. 127-141 http://www.jot.fm/issues/issue_2007_10/article5/

considered to be the most powerful support possible for testing a given hypothesis. In this paper, we empirically investigate and validate a set of OO metrics given by [Chidamber94] [Chidamber91] and [Braind99]. These metrics are analyzed by 12 software projects containing 136 classes. The study is divided into following parts:

(i) Principal component method of factor analysis is used to find whether all these metrics are independent or are capturing same underlying property of the object being measured.

(ii) Univariate logistic regression analysis is carried out to test the hypothesis that size, coupling and inheritance increase fault proneness of a class whereas cohesion increase decrease fault proneness of a class and find individual impact of metrics on fault proneness.

(iii) Finally a model using multivariate logistic regression analysis for predicting fault proneness of classes is given to predict which classes of a java application released in future will be faulty.

The results show that though the number of OO metrics is large but the number of dimensions actually found is much low. Further it was observed that import coupling (that count the number of other classes called by a class) metrics are strongly associated with fault proneness and predict faulty classes with high accuracy. Based on these results, it is reasonable to claim that such a model could help for planning and executing testing by focusing resources on fault prone parts of the design and code.

The paper is organized as follows: Section 2 summarizes the metrics studied, describes sources from which data is collected and presents hypothesis to be tested in the study. Section 3 presents the research methodology followed in this paper. In section 4 the results of the study are given. The model is evaluated in section 5. Limitations of the study are presented in section 6 and conclusions of the work are presented in section 7.

2 RESEARCH BACKGROUND

In this section, we present the summary of metrics studied in this paper (Section 2.1), empirical data collection (Section 2.2) and hypotheses to be tested in our work (Section 2.3). Our focus in the study is metrics proposed by [Chidamber94][Chidamber91] and [Braind99].

Metrics Studied

The metrics of coupling, cohesion, inheritance and size are the independent variables used in this study. Our focus is on OO metrics that are used as independent variables in a prediction model that is usable at early stages of software development. The metrics selected in this paper are summarized in Table 1. These metrics are explained with examples in [Aggarwal05][Aggarwal06].

Metric	Definition	Sources
Coupling between Objects (CBO)	CBO for a class is count of the number of other classes to which it is coupled.	[Chidamber94]
Coupling between Objects (CBO1)	Same as CBO, except that inheritance based coupling is not counted.	[Chidamber91]
Lack of Cohesion (LCOM1)	It counts number of null pairs of methods that do not have common attributes.	[Chidamber91]
Lack of Cohesion (LCOM2)	It measures the dissimilarity of methods in a class by looking at the instance variable or attributes used by methods.	[Chidamber94]
Number of Children (NOC)	The NOC is the number of immediate subclasses of a class in a hierarchy.	[Chidamber94]
Depth of Inheritance (DIT)	The depth of a class within the inheritance hierarchy is the maximum number of steps from the class node to the root of the tree and is measured by the number of ancestor classes.	[Chidamber94]
Weighted Methods per Class (WMC)	The WMC is a count of sum of complexities of all methods in a class.	[Chidamber94]
Response for a Class (RFC)	The response set of a class (RFC) is defined as set of methods that can be potentially executed in response to a message received by an object of that class.	[Chidamber94]
IFCAIC ACAIC OCAIC FCAEC DCAEC OCAEC IFCMIC ACMIC DCMIC FCMEC DCMEC OCMEC IFMMIC AMMIC OMMIC FMMEC DMMEC OMMEC	These coupling metrics count number of interactions between classes. The metrics distinguish the relationship between the classes (friendship, inheritance, none), different types of interactions, and the locus of impact of the interaction. The acronyms for the metrics indicates what interactions are counted: • The first or first two characters indicate the type of coupling relationship between classes (A: Ancestor, D: Descendents, F: Friend classes, IF: Inverse Friends (classes that declare a given class a as their friend), O: Others, i.e., none of the above relationships). • The next two characters indicate the type of interaction: CA: There is a Class-Attribute interaction if class x has an attribute of type class y. CM: There is a Class-Method interaction if class x consist of a method that has parameter of type class y. MM: There is a Method-Method interaction if class x calls method of another class y, or class x has a method of class y as a parameter. • The last two characters indicate the locus of impact:	[Braind99]

	IC: Import coupling, counts the number of other classes called by class x. EC: Export coupling, count number of other classes using class y.	
Lines Of Code (LOC)	It is the count of lines in the text of the source code excluding comment lines	

Table 1: Object-Oriented Metrics

Empirical Data Collection

To analyze the metrics chosen for this work, their values are computed for twelve different systems. These systems are developed by undergraduate engineering students and Masters of Computer Application students at School of Information Technology, of our University. The systems were developed using Java programming language over duration of four months. The aim was to teach the students system analysis and design techniques as part of their course curriculum. All students had experience with Java language and thus they had basic knowledge necessary for this study. The students were also taught about algorithmic detail

The students were divided into 12 teams of four students each. Each team developed a medium-sized system such as flight reservation, chat server, proxy server etc. The development process used was waterfall model. Documents were produced at each phase of software development. Faults were reported to the developers. A separate group of students having prior knowledge of system testing under the guidance of senior faculty were assigned the task of testing systems according to test plans.

The following relevant data was collected:

1. The design and source code of the java programs
2. The faulty data found by the testing team.

The 12 systems under study consist of 136 classes (39 KLOC) out of which 85 are system classes and 51 standard library classes available in java language. These classes contain functions to manipulate files, strings, lists, hash tables, frames, windows, menus, threads, socket connection etc.

All metric values are computed on system classes whereas coupling and inheritance metrics are also calculated between ‘system classes’ and ‘standard library classes’. It was observed during testing that the classes coupled with standard library classes were less fault prone than those coupled with system classes. It was also noticed that a large number of system classes inherited standard library classes. These classes did not need much testing as compared to the system classes, which inherit some of other system classes. Thus, the values of metrics for standard library classes are separately shown, as their effect on fault proneness is different from system classes.

Hypotheses

We test the hypotheses given below to find our empirical consequences.

H1 (for import coupling metrics): A class with more import coupling than its peers is more fault-prone as compared to them. (Null hypothesis: A class with more import coupling than its peers is less fault-prone prone as compared to them).

H2 (for export coupling metrics): A class with more export coupling than its peers is more fault-prone as compared to them. (Null hypothesis: A class with more export coupling than its peers is less fault-prone as compared to them).

H3 (for cohesion metrics): A class with lower cohesion than its peers is more fault-prone as compared to them. . (Null hypothesis: A class with lower cohesion than its peers is less fault-prone as compared to them).

H4 (for DIT metric): A class located lower in a class inheritance hierarchy than its peers is more fault-prone as compared to them. (Null hypothesis: A class located lower in a class inheritance hierarchy than its peers is less fault-prone as compared to them.).

H5 (for NOC metric): A class with a larger number of descendants than its peers is more fault-prone as compared to them. (Null hypothesis: A class with a larger number of descendants than its peers is less fault-prone as compared to them).

H6 (for size metrics): A class with a larger size i.e. more information than its peers is more fault-prone as compared to them. (Null hypothesis: A class with a larger size i.e. more information than its peers is less fault-prone as compared to them).

3 RESEARCH METHODOLOGY

In this section, the procedure used to analyze the data collected for each measure is described in following stages:

1. Principal-Component Method: Principal-Component Method (or P.C. method) is used to maximize the sum of squared loadings of each factor extracted in turn. The P.C. method aims at constructing new variable (P_i), called Principal Component (P.C.) out of a given set of variables $X_j's (j = 1,2,....,k)$.

   The variables with high loadings help identify the dimension P.C. is capturing, but this usually requires some degree of interpretation. In order to identify these variables, and interpret the P.C.s, we consider the rotated components. As the dimensions are independent, orthogonal rotation is used. There are various strategies to perform such rotation. We used the varimax rotation, which is the most frequently used strategy in literature. Eigenvalue (or latent root) is associated with each P.C. It refers to the sum of squared values of loadings relating to dimension, and then the sum is referred to as eigenvalue. Eigenvalue indicates the relative importance of each dimension for the particular set of variables being analyzed. In our study, the P.C.s with eigenvalue greater than 1 is taken for interpretation [Kothari89].
2. Logistic Regression (LR) and model prediction: LR is the most widely used technique [Hosmer89] in literature used to predict dependent variable from set of independent variables (a detailed description is given by [Basili96] and [Hosmer89]). In our work independent variable are OO metrics and dependent variable is fault proneness. LR is of two types: (i) Univariate LR (ii) Multivariate LR

   Univariate LR is a statistical method that formulates a mathematical model depicting relationship among each independent variable and dependent variable. This technique is used to test hypotheses given in Section 2.3.

   Multivariate LR is used to construct a prediction model for the fault-proneness of classes. In this method combination of metrics are used to determine the effect on dependent variable.

   In LR two stepwise selection methods forward selection and backward elimination are

used [Hosmer89]. In forward stepwise procedure, stepwise variable entry examines the variables in the block at each step for entry. The backward elimination method includes all the independent variables in the model. Variables are deleted one at a time from the model until a stopping criteria is fulfilled. We have used backward elimination method using metrics selected in P.C. method and univariate analysis.

For model prediction a test of multicollinearity is performed. The interpretation of model becomes difficult if multicollinearity is present. Let $X_1, X_2, ...X_n$ be the covariates of the model predicted. P.C. method is applied on these variables to find maximum eigenvalue, e_{max} and minimum eigenvalue, e_{min}. The conditional number is defined as $\lambda = \sqrt{e_{max}/e_{min}}$. If the value of the conditional number is 30 then multicollinearity is not tolerable [Belsley80].

The following statistics are reported for each significant metric:

- **Odds Ratio:** It is the probability of the event divided by the probability of the non-event. The event in our study is having a fault and nonevent is probability of not having a fault.
- **Maximum Likelihood Estimation (MLE) and Coefficients (A_i's):** MLE is a statistical method for estimating the coefficients of a model. The likelihood function (L) measures the probability of observing the set of dependent variable values (P_1, $P_2 \ldots P_n$). .
- **The statistical significance (*sig*):** It is the significance level of the coefficient, larger the statistical significance less is the estimated impact of the independent variables (OO metrics). In our study we used 0.05 as the significance threshold.
- **The R^2 Statistic:** It is the proportion of the variance in the dependent variable that is explained by the variance of the independent variables. The higher the effect of the model's explanatory variables implies better accuracy of the model.

3. Performance Evaluation: The model is evaluated in following ways:
   - The sensitivity and specificity of the model is calculated to predict the correctness of the model. The percentage of classes correctly predicted to be fault prone is known as sensitivity of the model. *Sensitivity* can be formally defined as:

$$Senstivity = \frac{\text{Classes correctly predicted as fault prone}}{\text{Classes actually fault prone}} \times 100 \quad (1)$$

The higher the sensitivity (% correct predictions), the better the model. The percentage of non-occurrences correctly predicted i.e. classes predicted not to be fault prone is called specificity of the model.

Specificity can be formally defined as:

$$Specificity = \frac{\text{Classes correctly predicted not to be fault prone}}{\text{Classes actually not fault prone}} \times 100 \quad (2)$$

Ideally both the sensitivity and specificity should be high. A low sensitivity means that there are many low risk classes that are classified as faulty. Therefore, the organization would waste resource in focusing

additional testing effort on these classes. A low specificity means that there are many high risk classes that are classified as not faulty. Therefore, the organization would be passing high risk classes to customers.

- To predict the accuracy of model it should be applied on different data sets. Therefore we performed k-cross validation of model [Stone74]. The data set is randomly divided into k subsets. Each time one of the k subsets is used as the test set and the other k-1 subsets are used to form a training set. Thus we get the fault proneness for all the k classes.

4 ANALYSIS RESULTS

This section presents the analysis results, following the procedure described in Section 3. P.C. analysis (Section 4.1), univariate analysis (Section 4.2) and multivariate analysis (Section 4.3) results are presented.

Principal Component (P.C.) Method

The coupling of system classes to system classes is counted separately from coupling of system classes to standard library classes. SL is suffixed with the metric name when coupling to standard library classes is counted. For instance CBO metric in such case is named as CBO_SL. The P.C. extraction method and varimax rotation method is applied on all metrics. The rotated component matrix is given in Table 2. The values above 0.7 (shown in bold in Table 2) are the metrics that are used to interpret the P.C.s. For each P.C., we also provide its eigenvalue, variance percent and cumulative percent. The interpretations of PCs are given as follows:

- P1: CBO_SL, OCAIC_SL, OCMIC_SL, CBO1_SL and OMMIC_SL measure coupling from standard library classes.
- P2: LCOM1, LCOM2, WMC and OCMIC. This dimension includes coupling, cohesion and size metrics. This indicates that import coupling and cohesion metrics have correlation with size.
- P3: OMMIC, RFC are coupling metrics. These metrics count import coupling from system classes through method invocations.
- P4: AMMIC_SL, OCAIC are import coupling metrics.
- P5: CBO, CBO1 are coupling metrics that count both import and export coupling.
- P6: NOC is an inheritance metric that counts number of children of a class.

Hence, we see that 5 out of 6 dimensions contain coupling metrics. Two dimensions P4 and P6 capture inheritance based coupling and inheritance metric. We also see that metrics capturing different properties are included in the same dimension P2.

Univariate Logistic Regression (LR) Analysis

In this subsection we find the relationship of independent variables (OO metrics) with dependent variable (fault proneness). Univariate LR analysis is done on 85 system classes. The table 3 provides the coefficient (B), standard error (SE), statistical significance (sig), R^2

statistic and odds ratio (exp(B)), for each measure. Metrics with no variance or lower variance are excluded from the table. The metrics with a significant relationship to fault proneness, that is, below or at the significance (named as Sig. in Table 3) threshold of 0.05 are shown in bold (see Table 3). The metrics that are not shown in bold do not have a significant relationship with fault proneness.

P.C.	P1	P2	P3	P4	P5	P6
Cumulative %	32.608	44.97	56.010	63.676	70.424	75.603
Variance%	32.608	12.3	11.03	7.665	6.748	5.1788
Eigenvalue	6.84	2.59	2.31	1.60	1.41	1.08
AMMIC_SL	0.04	-0.02	0.11	**0.77**	0.08	-0.14
CBO	0.12	0.00	0.18	0.14	**0.91**	-0.05
CBO_SL	**0.80**	0.15	0.07	0.37	0.12	0.16
CBO1	0.03	-0.03	0.18	-0.11	**0.94**	-0.01
DIT	-0.25	-0.14	0.36	-0.29	-0.28	-0.25
DIT_SL	0.15	-0.126	-0.13	0.51	-0.08	-0.08
LCOM1	0.28	**0.87**	0.26	0.06	-0.07	0.01
LCOM2	0.28	**0.88**	0.21	0.01	-0.08	0.00
LOC	0.27	0.41	0.68	0.02	0.05	0.17
WMC	0.35	**0.74**	0.49	0.16	0.01	0.17
NOC	0.10	-0.07	0.17	-0.04	-0.08	**0.80**
OCAEC	0.48	-0.00	-0.04	0.41	-0.09	-0.41
OCAIC	0.10	0.19	0.08	**0.74**	0.04	0.39
OCAIC_SL	**0.91**	0.15	0.00	-0.01	-0.02	-0.02
OCMIC	-0.04	**0.79**	-0.22	-0.17	0.11	-0.13
OCMIC_SL	**0.71**	0.46	0.21	0.15	0.04	-0.01
OCMEC	0.05	-0.45	0.25	-0.15	0.15	--0.02
OMMEC	0.11	-0.09	0.66	-0.10	0.25	-0.20
OMMIC	0.01	0.04	**0.74**	0.00	0.15	0.23
CBO1_SL	**0.80**	0.15	0.07	0.37	0.12	0.16
OMMIC_SL	**0.82**	0.09	0.38	-0.13	0.14	0.00
RFC	0.20	0.34	**0.76**	0.15	0.07	0.17

Table 2: Rotated Principal Component

The following observations are made based on the results given in Table 3:

- CBO and CBO1 metrics that count the both import and export coupling are related to fault proneness supporting hypotheses H1. Hence we reject the null hypothesis.
- But metrics OMMEC, OCMEC and OCAEC are not strongly related to fault proneness i.e. for instance if a classA is coupled to classB this will not make classB fault prone. Similar results have been shown in [Braind00]. Hence null hypothesis is accepted for export coupling metrics and hypothesis H2 is rejected.

- LCOM1 and LCOM2 metrics show positive coefficients. This indicates that the probability of fault proneness increases as the cohesion of a class decreases. Thus we accept the hypotheses H3 and reject null hypothesis.
- The results indicate that inheritance metric DIT measuring depth of inheritance tree is not related to fault proneness. This shows that student programmers give more attention to classes being inherited (i.e super classes) and follow a well-defined strategy. Hence null hypothesis is accepted for DIT metrics and hypothesis H4 is rejected.
- Metric NOC counting number of children of a class is not related to fault proneness. Hence null hypothesis is accepted for NOC metric and hypothesis H5 is rejected.

Metric	B	S.E.	Sig.	R^2	Exp(B)
CBO	0.8436	0.2802	**0.0026**	0.1206	2.3246
CBO1	0.6180	0.2491	**0.0131**	0.077	1.8553
CBO_SL	0.4696	0.1513	**0.0019**	0.112	1.5993
CBO1_SL	0.4696	0.1513	**0.0019**	0.112	1.5993
LCOM1	0.0612	0.0244	**0.0121**	0. 2155	0.0631
LCOM2	0.0800	0.0347	**0.0212**	0.1982	1.0832
DIT	-0.7518	0.4279	0.0789	0.0344	0.4715
NOC	0.3147	0.2666	0.2379	0.0172	1.3698
DIT_SL	-0.2760	0.7655	0.7185	0.0000	0.7588
LOC	0.0100	0.0033	**0.0025**	0.273	1.0101
RFC	0.1817	0.0410	**0.0000**	0.536	1.1993
WMC	0.2466	0.0646	**0.0001**	0.375	1.2796
OCAEC	0.0731	0.2552	0.7746	0.0000	1.0758
OCAIC	0.9381	0.3594	**0.0090**	0.077	2.5552
OCMEC	0.1956	0.2393	0.4138	0.0086	1.2160
OCMIC	0.2065	0.4001	0.6058	0.0000	1.2293
OMMEC	0.0358	0.0262	0.1721	0.0172	1.0364
OMMIC	0.458	0.1121	**0.000**	0.362	1.5809
OCAIC_SL	0.3240	0.1258	**0.0100**	0.0862	1.3827
OCMIC_SL	0.3170	0.1129	**0.0050**	0.1206	1.3730
OMMIC_SL	0.1754	0.0625	**0.0050**	0.1724	1.1917
AMMIC_SL	3.2265	10.5891	0.7606	0.0431	25.1906

Table 3: Univariate LR Analysis of Metrics

- Size metric WMC is related to fault proneness and thus hypothesis H6 is accepted.

Multivariate Logistic Regression (LR) Analysis

In this section we predict model to identify the faulty classes. Metrics are pre selected using results from P.C. and univariate analysis using backward elimination method. The model includes an intercept referred to as constant.

The model includes two coupling metrics OMMIC and RFC. One size metric WMC is also included in the model. OMMIC and RFC metrics were covered in dimension P3 and also

found strongly related to fault proneness in univariate analysis. WMC metric is captured in dimension P2. The summary of Model statistics is presented in Table 4. The conditional number is $\sqrt{3.781/0.03453} = 10.46$ that does not indicate any problem.

Variable	B	S.E.	Sig.
OMMIC	0.5668	0.1662	0.0006
WMC	0.3047	0.1140	0.0075
RFC	0.1046	0.0433	0.0156
Constant	-4.6962	1.0297	0.0000
-2 Log likelihood: 35.577			
R^2 Statistic: 0.7			

Table 4: Model Statistics

The model was applied to 85 system classes and accuracy of the model is presented in Table 5. The R^2 statistic and log likelihood of the model is fairly high. Out of 37 classes actually fault prone, 32 classes were predicted to be fault prone. The sensitivity of the model is 86.49%. Similarly 45 out of 48 classes were predicted not to be fault prone. Thus specificity of the model is 93.75%.

		Predicted	
		Not Faulty	Faulty
		Po<=0.5	Po>0.5
Observed	Not Faulty	45	3
	Faulty	5	32

Table 5: Predicted Correctness of Model

5 MODEL EVALUATION

The sensitivity and specificity of model predicted in previous section is quite high but it is somewhat optimistic since the model is applied on same data set from which it is derived from. To predict accuracy of model it should be applied on the different data sets. Thus we performed 9-cross validation of model following the procedure given in Section 3. For the 9-cross validation, the classes were randomly divided into 9 parts of approximately equal (5 partitions of 9 data points each and 4 partitions of 10 data points each).

		Predicted	
		Not Faulty	Faulty
		Po<=0.5	Po>0.5
Observed	Not Faulty	45	3
	Faulty	7	30

Table 6: Results of 9-cross validation of Model

Table 6 shows that 30 out of 37 classes are correctly predicted to be fault prone. The sensitivity of the model is 81.89%. Similarly 45 out of 48 classes were predicted not to be fault prone. Thus specificity of the model is 93.75%. This shows that the model also predicts classes with similar data set other than from which it is derived from with high accuracy.

6 THREATS TO VALIDITY

The study has a number of limitations that are not unique to our study but are common with most of the empirical studies in the literature. However, it is necessary to repeat them here.

The degree to which the results of our study can be generalized to other research settings is questionable. The reason is that the systems developed are small-sized. The developers are students and hence are not well trained as professional developers.

In this study the severity of faults is not taken into account. There can be number of faults which can leave the system in various states e.g. a failure that is caused by a fault may lead to a system crash or an inability to open a file. The former failure is more severe than latter, thus the types of fault is not the same. The same limitation is also reported in [Emam99].

Though these results provide guidance for future research on the impact of OO metrics on fault proneness, further validations are needed with different systems to draw stronger conclusions.

7 CONCLUSIONS

We have conducted an empirical validation of twenty six metrics. The systems under study are medium sized systems written in Java and have a testing record including number of faults found in each class. In this study we first find the interrelationships among selected metrics and then found the individual and combined effect of selected metrics on fault proneness.

The number of dimensions captured in P.C. analysis is only 6, which are much lower than the number of metrics. This simply supports the fact that many of the metrics proposed are based on comparable ideas and therefore provide somewhat redundant information.

The results of univariate LR analysis show that most of the import coupling and cohesion metrics are found related to fault proneness. On the other hand inheritance metrics were not found related to fault proneness.

The results of multivariate LR analysis show that import coupling and size metrics measure fault proneness with high accuracy. As far as cohesion metrics are concerned they were found highly related to fault proneness in univariate LR analysis but none was found significantly related to fault proneness in multivariate LR analysis. The model has sensitivity 86.5% and specificity above 90%.

The metrics could not be evaluated over a large data set but this is a problem that has plagued much of empirical software engineering research. More similar type of studies must be carried out with different data sets to give generalized results across different organizations. We plan to replicate our study on large data set and industrial OO software system. We further plan to predict the models based on early analysis and design artifacts.

REFERENCES

[Binkley98] A.Binkley and S.Schach, "Validation of the Coupling Dependency Metric as a risk Predictor", *International Conference on Software Engineering (ICSE)*, 452-455, 1998.

[Lake94] A.Lake, C.Cook, "Use of factor analysis to develop OOP software complexity metrics", *Proc. 6th Annual Oregon Workshop on Software Metrics, Silver Falls*, Oregon, 1994.

[Henderson96] B.Henderson-sellers, "*Object-Oriented Metrics, Measures of Complexity*", Prentice Hall, 1996.

[Kothari89] C.R.Kothari, "*Research Methodology. Methods and Techniques*", New Age International Limited.

[Belsley80] D.Belsley, E. Kuh, R. Welsch, "*Regression Diagnostics: Identifying Influential Data and Sources of Collinearity*", John Wiley & Sons, 1980.

[Hosmer89] D.Hosmer, S.Lemeshow, "*Applied Logistic regression*", John Wiley and Sons, 1989.

[Tegarden95] D.Tegarden, S. Sheetz, D.Monarchi, "A Software Complexity Model of Object- Oriented Systems", *Decision Support Systems*, vol. 13 no.3-4, 241-262, 1995.

[Bieman95] J.Bieman, B.Kang, "Cohesion and Reuse in an Object-Oriented System", *Proc. CM Symp. Software Reusability (SSR'94)*, 259-262, 1995.

[Emam99] K..El Emam, S. Benlarbi, N.Goel , S. Rai, "A Validation of Object-Oriented Metrics", Technical Report ERB-1063, *National Research Council of Canada (NRC)*, 1999.

[Emam01] K.El Emam, W. Melo, J. Machado, "The Prediction of Faulty Classes Using Object-Oriented Design Metrics", *Journal of Systems and Software*, vol. 56, 63-75, 2001.

[Aggarwal05] K.K.Aggarwal, Yogesh Singh, Arvinder Kaur, Ruchika Malhotra, "Analysis of Object-Oriented Metrics", International Workshop on Software Measurement (IWSM), Montréal, Canada , 2005.

[Aggarwal06] K.K.Aggarwal, Yogesh Singh, Arvinder Kaur, Ruchika Malhotra, "Empirical Study of Object-Oriented Metrics", *Journal of Object-Technology*, vol. 5, no. 8, 149-173, 2006.

[Aggarwal05] K.K.Aggarwal, Yogesh Singh, Arvinder Kaur, Ruchika Malhotra, "Software Reuse Metrics for Object-Oriented Systems", *Third ACIS Int'l Conference on Software Engineering Research, Management and Applications (SERA'05)*, IEEE Computer Society, 48-55, 2005.

[Braind98] L.Briand, J.Daly and J. Wust, "A Unified Framework for Cohesion Measurement in Object-Oriented Systems", *Empirical Software Engineering*, 3, 65-117, 1998.

[Braind99] L.Briand , J.Daly and J. Wust, "A Unified Framework for Coupling Measurement in Object-Oriented Systems", *IEEE Transactions on software Engineering*, vol. 25, 91-121, 1999.

[Braind00] L.Briand , J.Daly, V.Porter, J. Wust, "Exploring the relationships between design measures and software quality", *Journal of Systems and Software*, vol. 5, 245-273, 2000.

[Braind01] L. Briand, J. Wüst, H. Lounis, "Replicated Case Studies for Investigating Quality Factors in Object-Oriented Designs, *Empirical Software Engineering: An International Journal*, vol 6, no 1, 11-58, 2001.

[Cartwright00] M.Cartwright, M.Shepperd, "An Empirical Investigation of an Object-Oriented Software System", *IEEE Transactions of Software Engineering*. vol.26, Issue 8, 786 – 796, Aug. 2000.

[Hitz00] M.Hitz, B. Montazeri, "Measuring Coupling and Cohesion in Object-Oriented Systems", *Proc. Int. Symposium on Applied Corporate Computing*, Monterrey, Mexico, 1995.

[Lorenz94] M.Lorenz, J.Kidd, " *Object-Oriented Software Metrics*", Prentice-Hall, 1994.

[Stone74] M.Stone, "Cross-validatory choice and assessment of statistical predictions", *J. Royal Stat. Soc*., 36, 111-147, 1974.

[Harrison98] R.Harrison, S.J.Counsell, R.V.Nithi, "An Evaluation of MOOD set of Object-Oriented Software Metrics", *IEEE Trans. Software Engineering*, vol. SE-24, no.6, 491-496, 1998.

[Chidamber94] S.Chidamber and C.Kemerer, "A metrics Suite for Object-Oriented Design", *IEEE Trans. Software Engineering*, vol. SE-20, no.6, 476-493, 1994.

[Chidamber91] S.Chidamber, C. Kemerer, "Towards a Metrics Suite for Object Oriented design", *Proc. Conference on Object-Oriented Programming: Systems,*

Languages and Applications (OOPSLA'91), Published in SIGPLAN Notices, vol 26 no. 11, 197-211, 1991.

[Chidamber98] S.Chidamber, D. Darcy, C. Kemerer, "Managerial use of Metrics for Object-Oriented Software: An Exploratory Analysis", *IEEE Transactions on Software Engineering*, vol.24, no.8, 629-639, 1998.

[Gyimothy05] T.Gyimothy, R. Ferenc I. Siket, **"**Empirical validation of object-oriented metrics on open source software for fault prediction", *IEEE Trans. Software Engineering*, vol. 31, Issue 10, 897 – 910, Oct. 2005.

[Basili96] V.Basili, L.Briand, W.Melo, "A Validation of Object-Oriented Design Metrics as Quality Indicators", *IEEE Transactions on Software Engineering*, vol. 22 no.10, 751-761, 1996.

[Li93] W.Li, S.Henry, "Object-Oriented Metrics that Predict Maintainability', *Journal of Systems and Software*, vol. 23, no.2, 111-122, 1993.

[Lee95] Y.Lee, B.Liang, S.Wu, F.Wang, "Measuring the Coupling and Cohesion of an Object-Oriented program based on Information flow", *International Conference on Software Quality*, Maribor, Slovenia 1995.

[Ping02] Yu Ping, Ma Xiaoxing, Lu Jian "Predicting Fault-Proneness using OO Metrics: An Industrial Case Study, *CSMR 2002*, Budapest, Hungary, 99-107.

About the authors

K. K. Aggarwal is vice chancellor at the Guru Gobind Singh Indraprastha University, India. He received his doctorate from Kurushetra University. He was president of the Institution of Electronics and Telecommunication Engineers (IETE) from 2002 through 2004. Recently he was awarded "Delhi Ratan Bhuddhijeevi Samman" by the All India Conference of Intellectuals (AICI). Prof. Aggarwal has written few books and many of his articles have appeared in several books published by IEEE of USA. He is coauthor of a book on software engineering and has published more than 300 publications in national and international journals and conferences. He can be reached by e-mail at kka@ipu.edu.

Yogesh Singh is a professor with the University School of Information Technology, Guru Gobind Singh Indraprastha University, Kashmere Gate, India. He is also controller of examination with Guru Gobind Singh Indraprastha University, Kashmere Gate, India. He received his master's degree and doctorate from the National Institute of Technology, Kurukshetra. His area of research is Software Engineering focusing on Planning, Testing, Metrics and Neural Networks. He is coauthor of a book on software engineering, and is a Fellow of IETE and member of IEEE. He has more than 150 publications in international and national journals and conferences. Singh can be contacted by e-mail at ys66@rediffmail.com.

Arvinder Kaur is a Reader with the University School of Information Technology. She obtained her doctorate from Guru Gobind Singh Indraprastha University and her master's degree in computer science from Thapar Institute of Engg. and Tech. Her research interests include software engineering, object-oriented software engineering, software metrics, microprocessors, and operating systems. She is also a lifetime member of ISTE and CSI. Kaur has published more than 30 research papers in national and international journals and conferences. Her paper titled "Analysis of object oriented Metrics" was published as a chapter in the book *Innovations in Software Measurement* (Shaker -Verlag, Aachen 2005). She can be reached by e-mail at arvinderkaurtakkar@yahoo.com.

Ruchika Malhotra is a research scholar with the University School of Information Technology, Guru Gobind Singh Indraprastha University, India. She received her master's degree in software engineering from the University School of Information Technology, Guru Gobind Singh Indraprastha University, India. Her research interests are in improving software quality, statistical and adaptive prediction models for software metrics, neural nets modeling, and the definition and validation of software metrics. She has more than 10 publications in international journals and conferences. Her paper titled "Analysis of object oriented Metrics" was published as a chapter in the book *Innovations in Software Measurement* (Shaker -Verlag, Aachen 2005). She can be contacted by e-mail at ruchikamalhotra2004@yahoo.com.