Detection of Fault-Prone Classes Using Logistic Regression Based Object-Oriented Metrics Thresholds

Shahid Hussain, Jacky Keung, Arif Ali Khan, Kwabena Ebo Bennin
Department of Computer Science
City University of Hong Kong
Shussain7-c@my.cityu.edu.hk, Jacky.Keung@cityu.edu.hk, [aliakhan2-c, kebennin2-c]@ my.cityu.edu.hk

***Abstract*—Background: In the plethora of studies, the object-oriented metrics have been empirically validated to assess the design properties and quantify the high-level quality attributes such as fault-proneness, either at the method or class granularity levels of software. Motivation: A more precise value of an object-oriented metric can be used as an indicator for the developers to make the informed decisions regarding the detection of design flaws and classify the fault-proneness classes. *Method*: Bender used an approach in the domain of epidemiology studies to derive the threshold values for the risk factors. In our study, we follow the Bender's approach and propose a model to derive the thresholds for a set of software design metrics via non-linear functions, which are described through logistic regression coefficients. Subsequently, we perform four types of analysis and three experiments in order to evaluate and compare the effectiveness of derived thresholds in the domain of classification of fault proneness classes. We use the Precision, Recall, F-measure and classification accuracy performance measures to assess the effectiveness of derived metrics thresholds. *Results*: We compare the derive threshold values of DIT, CA, LCOM and NPM metrics with their existing data distribution based threshold values, and observed the significant increase in the classification accuracy of fault-prone classes. For example, DIT (27%), Ca (2%), NPM (2%) and LCOM (15%) for the Ant-1.5 project. *Conclusion*: The analysis results suggest that the proposed model can be applied to derive the thresholds of other object-oriented metrics which present either with or without heavy-tailed distribution, however, the proposed model to derive thresholds cannot generalize for all the systems due to variation in data characteristics.**

Keywords*—Object-Oriented Metrics; Logistic Regression; Fault Proneness; Complexity; Classification; Performance Measures.

I. INTRODUCTION

Software metrics can be used for many purposes to aid developers in their decision making and monitoring the quality of projects. Through a set of one or more than one software metrics, developers can measure an internal attribute of a software system, and enabled its relation with high-quality attributes such as fault-proneness. However, there are some variations in how the software metrics are defined and their values are collected [2]. Nnumerous software metrics suit have been proposed and empirically validated, to predict and investigate the high-level quality attributes of design entities. The most common metrics suits are Chidamber and Kemerer (CK) design metric suite [3], MOOD metric suite [4], and QMOOD metric suite [1]. Moreover, these metrics have been used in software quality auditing tools which are available either commercially or as an open source, such as JHawk and Ckjm.

Subsequently, the structure of object-oriented software includes a set of design entities such as classes, methods and packages with their relationships, such as inheritance and composition. The object-oriented software metrics are used to compute the complexity of design entities and enable its relationship with external quality attributes such as to scrutinize the fault-prone classes. For example, J. Bansiya and C. Davis describes a hierarchical model for the assessment of high-level quality attributes such as flexibility and reusability using a suite of object-oriented design metrics[1]. Moreover, In the domain of fault prediction, 49% object-oriented metrics are used in empirical studies as compared to traditional source metrics [6]. The threshold values of these metrics can help the developer and the tester to take informed decisions regarding classification of fault proneness classes. Commonly, the threshold values of software metrics for the interpretation of software quality are defined either through developer experience[7], data distribution [8] or by using different statistical techniques [9-12]. In our study, we follow the Bender's approach and propose a model to derive the threshold values for a set of object-oriented design metrics, and evaluate its effect in order to detect and classify the fault-prone classes. The set of 13 metrics includes five J. Bansiya metrics [1] that is Direct Access Metric (DAM), Cohesion Among Method of Class (CAM), Measure of Aggregation (MOA), Number of Public Methods (NPM) and Measure of Functional Abstraction(MFA), six Chidamber and Kemerer design metrics[3] that is Weighted Method per Class (WMC), Depth of Inheritance Tree (DIT), Coupling Between Objects (CBO), Response For a Class (RFC), Number Of Children (NOC) and Lack of Cohesion of Methods (LCOM) and two Martin's metrics [13] Afferent couplings (Ca) and Efferent couplings (Ce). Moreover, we perform four types of analysis shown in Figure 1 and three experiments in order to evaluate and compare the effectiveness of derived thresholds.

In the first analysis, we used the proposed model and derive metrics threshold values of two open source projects called JEdit-4.0 and Ant-1.3. Subsequently, in the second analysis, we use subsequent releases of JEdit-4.0 (that are JEdit-4.1, JEdit-4.2 and JEdit-4.3) and Ant-1.3 (that are Ant-1.4, Ant-1.5, Ant-1.6, and Ant-1.7) in order to evaluate the effectiveness of derived metrics thresholds in the domain of classification of fault-prone classes.

978-1-5090-3713-1/16 $31.00 © 2016 IEEE
DOI 10.1109/QRS-C.2016.16

Authorized licensed use limited to: CITY UNIV OF HONG KONG. Downloaded on September 17,2026 at 02:35:39 UTC from IEEE Xplore. Restrictions apply.

We used Precision, Recall, F-measure and classification accuracy performance measure in the evaluation process. In the third analysis, we compare the effectiveness of derived metrics thresholds with an existing data distribution based threshold values [8, 15]. Moreover, in third analysis, we also consider threshold effect of correlated metrics, such as CBO and DIT [23] correlation have a significant effect on the prediction of defects. Finally, we perform fourth analysis in order to generalize the applicability of the proposed model to derive metrics thresholds. In this analysis, we consider 31 open source projects retrieved from PROMISE repository. The descriptive statistics of studied open source projects and the subsequent releases of JEdit-4.0 and Ant-1.3 are shown in Table 1.

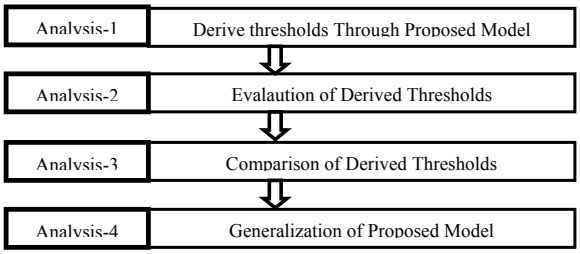

Figure 1. Logistic Curve and VARL for WMC of Berek

II. RELATED WORK

There are numerous software design metrics suites have been defined and empirically validated such as J. Bansaya [1], Chidamber and Kemerer [3] and Martin metrics suit [13], however, there is room for researchers to find a precise metric value which can help the developers to characterize and evaluate the software design, to detect design flaws [17], and identify the problematic part of the software. The software development community can use the software metrics in practice if they know the software design metrics thresholds, however, precise values for certain software metrics are not known, and one reason may be the lack of practical use due to their heavy tailed data distribution nature [8]. L.H. Rosenberg [7] applied and interpreted the object-oriented metrics by focusing on an approach used by Software Assurance Technology Center (SATC) at NASA Goddard Space Flight Center. In her study, Rosenberg used histograms to analyze the CK metrics and interpret its threshold affects over the external quality attributes. According to NASA approach, the selection of associated attributes of object-oriented software development should be prior to choosing metrics with its related threshold values.

Erni and Lewerentz [15] identify the threshold value T for a metric via mean (μ) and standard deviation (σ) parameters for the data distribution of a corresponding metric; that is $T=\mu \pm \sigma$. The values of a metric lower than $T=\mu - \sigma$ and greater than $T=\mu + \sigma$ refer to a problem. The effectiveness of this statistical technique for threshold calculation is related to the data which is normally distributed. Similarly, V. A. French [16] also use the mean (μ) and standard deviation (σ) parameters besides Chebyshev's inequality theorem to calculate the threshold T of a metric; that is $T=\mu + k * \sigma$, where k is the number of standard deviations. The validity of this statistical technique is not limited to normal distribution. In their study, S. Benlarbi et al [11] use a subset of CK metrics and investigate the relation of its thresholds with software failure. Moreover, S. Benlarbi compared two error probability models that are one model is without and one is with metrics thresholds. Subsequently, they reported the zero probability of errors when the metric value is below the threshold and also concluded that there is no significance difference between two models. Similarly, El Emam et al [12] also address in their comparative study that there is no optimal class size, and there is no empirical evidence for the use of metrics thresholds in order to predict the faults. Shatnawi et al. [17], have investigated the use of Receiver-Operating Characteristic (ROC) method in order to identify the thresholds for Chidamber and Kemerer (CBO, RFC, WMC, LCOM, DIT, and NOC), Lorenz and Kidd (NOAM, NOOM, NOA, and NOO) and Li (CTA and CTM) and use these threshold to predict the existence of bugs in different considered error categories. The objective of their study is to identify the thresholds to present a relationship between object-oriented metrics and error-severity categories. K.A. Ferreira [8] identify thresholds for six object-oriented metrics and rank the derived thresholds (other than with typical value) into good, regular and bad based on the frequency distribution using a statical approach. In their study, Ferreira et al, consider a large collection of open source programs developed in Java and performed four types of analysis with respect to entire dataset, software system sizes, application domain, and by type of software. Finally, they concluded that investigated metrics are modeled by a heavy-tailed distribution which means typical values for these metrics cannot be found, and there is need of reference values for these metrics.

R. Shatnawi [10] derive the threshold values for CK metrics via investigating the effect of log transformation on the data analysis, however, he did not consider the data distribution and skewness in the data. He used metrics thresholds to identify the fault-prone areas of 41 releases of 11 products. He conducts an experiment for fault-proneness classification with and without log transformation thresholds and obtains outperform results. Similarly, R. Shatnawi [9] use a statistical model based on logistic regression to calculates the threshold values of CK metrics. Initially, his proposed method was introduced by R. Bender [18] in a quantitative study in the epidemiological field to investigate the threshold effect for the quantification of risk associated with the factors. In his study, R. Shatnawi calculates the metrics thresholds with four risk level on Eclipse 2.0 and assess the thresholds effectiveness on Eclipse 2.1. Moreover, R. Shatnawi concluded that metrics thresholds at risk level 0.65 are effective, however at the same risk level, he cannot find the effectiveness of metrics thresholds for Mozilla and Rhino system. In our study, we also follow the Bender's approach to derived the threshold values of 13 object-oriented metrics. Our work is different from R. Shatnawi [9] in three aspects. Firstly, we also identify and assess the thresholds of those metrics which are not applied widely in industry, secondly, we derive metrics thresholds of two open source system JEdit-4.0 and Ant-1.3, and evaluate the effectiveness of derived thresholds through their subsequent

Authorized licensed use limited to: CITY UNIV OF HONG KONG. Downloaded on September 17,2026 at 02:35:39 UTC from IEEE Xplore. Restrictions apply.

releases. Finally, we derived metrics thresholds of 31 open source systems in order to investigate that how proposed model is applicable for different systems.

III. Proposed Model on Bender's Approach

In the domain of epidemiology study, R. Bender [18] introduce a model to derive threshold of risk factors and quantitatively assess the associated risks, such as glycosylated hemoglobin (HbA) threshold effect on the risk of microvascular diabetic complications. In our study, we follow R. Bender's approach and propose a model to derive the threshold values for a set of object-oriented metrics. R. Bender's approach introduce this model to overcome the deficiencies in the definition of logistic regression based threshold model, which was proposed by Ulm [19]. Moreover, El Emam [12] and Benlarbi [11] have followed the Ulm model to estimate the threshold values of object-oriented metrics, however, they observe no significant difference between the two models that is with or without thresholds.

The uniqueness of the proposed model is in its simplicity and the use of the logistic curve to estimate the threshold for a metric. The logistic curve of model presents the association between a continuous risk factor (that is object-oriented metrics) and a binary response variable (Fault-prone label for each class). Moreover, through the use of logistic curve, two types of non-linear functions of logistic regression coefficients are described to identify the threshold values of risk factors, which can help to quantify the risk that has a high fault probability in an event. In our study, the term event refers to either a fault-prone or not fault-prone class. The threshold values of metrics are defined at 95% confidence interval [20]. The equation 1 describes the general logistic regression model as follows.

$$p(x) = \frac{\exp(\alpha + \beta x)}{1 + \exp(\alpha + \beta x)} \tag{1}$$

Where the term p refers to the probability of a fault-prone class and describe as p(x)= P(Y=1 | Y=x). The term Y is a dependent variable which refers to the label of a class that is, either it is fault proneness or not. The terms x, α and β are object-oriented metric, estimated constant and estimated coefficient respectively. The first non-linear function as a benchmark can be defined as the Value of an Acceptable Risk Level (VARL) with a suggested probability P_0 (e.g. P_0=0.01 or P_0=0.05 or P_0=0.1). The equation 2 is used to describe the VARL non-linear function.

$$VARL = p^{-1}(P_0) = \frac{1}{\beta}(\log(\frac{P_0}{1-P_0}) - \alpha) \tag{2}$$

It is assumed that the values of metrics below its VARL threshold for a class refer to its lower risk of fault occurrence than a suggested probability (P_0). However, the shape of the relation between p and x that is when values of metrics are above the VARL then the risk of fault occurrence is above the suggested probability (P_0), but it has no practical relevance. In order to describe the shape of the response logistic curve, Bender takes the derivative of the logistic function (equation 1), which is shown in equation 3.

$$d(x) = p'(x) = \frac{\beta \exp(\alpha + \beta x)}{(1 + \exp(\alpha + \beta x))^2} \tag{3}$$

Equation 3 represents a unimodal and a symmetrical curve with maximum value d_{max} = β/4 at x_{max}= -α/β, which shows for any given value d_0 (except of d_{max} there exist two corresponding x values that is one below and one above d_{max}).

Table 1. Descriptive Statistics of Projects

Project Name	System Size (# of Classes)	Faulty Classes (#)
Berek	043	016
e-learning	064	005
Forrest-0.8	032	002
Intercafe	027	004
Kalkulator	027	006
Nieruchomosci	027	010
Pbeans2	051	010
Pdftranslator	033	025
Serapion	045	009
Skarbonka	045	009
Szybkafucha	025	014
termproject	042	013
workflow	039	020
wspomaganiepi	018	012
zuzel	029	013
sklebagd	020	012
systemdata	065	009
Ant-1.3	125	020
Camel-1.6	965	188
Ivy-2.0	352	040
JEdit-4.0	306	075
Log4j-1.2	205	189
lucene-2.4	340	203
Poi-3.0	442	281
Prop-6	660	066
redaktor	176	027
tomcat	858	077
velocity-1.6	230	079
xalan-2.7	909	898
xerces-1.4	588	437
synapse-1.2	256	086
Jedit-4.1	312	079
Jedit-4.2	367	048
Jedit-4.3	492	011
Ant-1.4	178	040
Ant-1.5	293	032
Ant-1.6	351	092
Ant-1.7	745	166

The second non-linear function as a benchmark can be defined as a lower Value of an Acceptable Risk Gradient (VARG) is shown in equation 4.

$$VARG = d^{-1}(d_0) = \frac{1}{\beta}(\log(\frac{\beta - 2d_0 - \sqrt{\beta^2 - 4d_0\beta}}{2d_0}) - \alpha) \tag{4}$$

Where the acceptable risk gradient is given by a value $d_0 \leq \beta/4$ (e.g. d_0 = 0.2). Moreover, if the chosen gradient d_0 is larger than β/4 then no corresponding VARG exist. This assumption remains true during our analysis, consequently, we only calculate the VARL for each metric and evaluate its effectiveness to detect and classify the fault proneness classes. It has been assumed in data analysis studies that statistical techniques which are assuming

Authorized licensed use limited to: CITY UNIV OF HONG KONG. Downloaded on September 17,2026 at 02:35:39 UTC from IEEE Xplore. Restrictions apply.

normal distribution may invalidate due to use of metrics which are presented in other than the normal distribution.

IV. Implementation Procedure

The implementation procedure of our proposed model is shown in the following subsections.

A. Dataset Analysis

We validate our proposed model on the collected metrics data of 31 open source projects, four releases of JEdit and five releases of Ant open source projects [14]. The authors have collected fault data using Buginfo, which analyze the history of classes through Subversion or Concurrent Versions System (CVS). We used the datasets to derive the metrics thresholds through proposed model except the subsequent releases of JEdit-4.0 (JEdit-4.1, JEdit-4.2, JEdit-4.3) and Ant-1.3 (Ant-1.4, Ant-1.5,Ant-1.6, Ant-1.7). We used datasets of subsequent releases of JEdit-40 and Ant-1.3 to evaluate the effectiveness of derived metrics thresholds in the domain of classification of fault-prone classes. The descriptive statistics of all open source projects are shown in Table 1.

B. Selected Metrics

In this study, we consider a set of 13 object-oriented software design metrics. There are two reasons for the selection of these metrics in our study, first is their frequent use in fault prediction studies and the second is their interpretation to detect the design problems, and enable the relationship between design properties and external quality attributes [1, 6]. The brief description of each object-oriented metric is as follows.

- **Weighted Methods per Class (WMC):** The value of WMC metric is equal to the number of all defined methods inside a class if it is assumed that each method is of similar complexity, otherwise WMC is a count of sum complexities of all methods defined in a class.
- **Depth of Inheritance Tree (DIT):** The value of DIT metric for a class is equal to the length of its longest path of a super class, from which it can inherit methods.
- **Coupling Between Objects (CBO):** The value of CBO metric for any specific class is equal to a number of other classes, to which that specific class is coupled.
- **Response For a Class (RFC):** The value of RFC metric is equal to a number of methods that could be executed in response to a message received by an object of a class.
- **Number Of Children (NOC):** The value of NOC metric for a class is equal to the number of immediate subclasses in the class hierarchy.
- **Lack of Cohesion of Methods (LCOM):** The value of LCOM metric for a class is equal to the difference between the number of methods-pairs with or without similarity in a sharing of member variables by the methods.
- **Number of Public Methods (NPM):** This metric simply counts all the methods in a class, which are declared as public.
- **Data Access Metric (DAM):** This metric refers to the ratio of the number of private (protected) attributes to the total number of attributes declared in the class.
- **Measure of Aggregation (MOA):** This metric helps to measure the extent of the part-whole relationship, realized by using attributes.
- **Measure of Functional Abstraction (MFA):** This metric is the ratio of the number of methods inherited by a class to its total number of methods accessible by the member methods
- **Afferent couplings (Ca):** The Ca metric represents the number of classes that depend upon the measured class.
- **Cohesion Among Methods (CAM):** This metric computes the relatedness among methods of a class based upon the parameter list of the methods.

Table 2. VARL Thresholds for JEdit-4.0 and Ant-1.3

Metric	JEdit-4.0	Ant-1.3
WMC	25.14	22.37
DIT	4.93	4.39
NOC	6.06	26.44
CBO	16.62	36.72
RFC	49.48	58.87
LCOM	603.71	598
CA	12.22	44.72
CE	8.64	1.3
NPM	11.51	23.83
DAM	0.75	1.29
MOA	1.45	2.72
MFA	0.93	0.90
CAM	0.75	0.67

C. First Analysis

In our first analysis, we compute the metrics thresholds using the proposed model. In this regard, we follow the procedure described in Section III and derive the VARL (Value of An Acceptable Risk Level) value of each metric (defined in Section IV-B) of two open source projects called Jedit-4.0 and Ant-1.3.

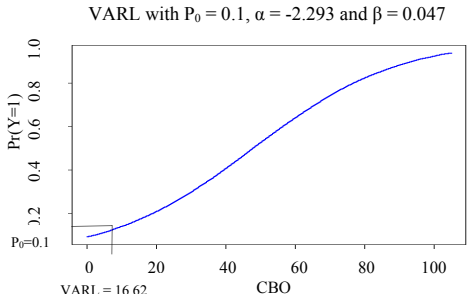

Figure 2. Logistic Curve and VARL for CBO of JEdit-4.0

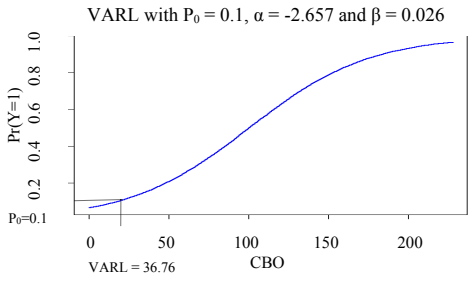

Figure 3. Logistic Curve and VARL for CBO of Ant-1.3

Authorized licensed use limited to: CITY UNIV OF HONG KONG. Downloaded on September 17,2026 at 02:35:39 UTC from IEEE Xplore. Restrictions apply.

Moreover, we cannot find VARG (Value of Acceptable Risk Gradient) values of these metrics due to chosen gradient d_0 (=0.2) > β/4. The computed VARL threshold values are shown in Table 2. The logistic curve for the computed VARL threshold values (16.62 and 36.72) of CBO metric for JEdit-4.0 and Ant-1.3 are shown in Figure 2 and Figure 3 respectively.

D. Second Analysis

Like other domain (such as medicine), in software engineering, metrics thresholds can help the specialist in evaluating the software process, and assure the products relevant quality attributes. Moreover, we consider the subsequent release of JEdit-4.0 and Ant-1.3 to evaluate the effectiveness of their metrics thresholds. In the second analysis, we evaluate the effectiveness of metric thresholds which are derived from the proposed model. In this regard, we performed our first experiment and evaluate the accuracy of derived metrics thresholds in order to classify the fault proneness classes. In order to perform this experiment, we follow the steps given below.

Setp-1. We collect the faulty data of subsequent releases of JEdit-4.0 (that are JEdit-4.1, JEdit-4.2 and JEdit-4.3) and Ant-1.3 (that are Ant-1.4, Ant-1.5, Ant-1.6, and Ant-1.7) from the open source project organized by PROMISE repository [14].

Step-2. We label the classes (of each subsequent release of JEdit-4.0 and Ant-1.3) as fault-prone if they are associated with at least one bug otherwise label it as not fault-prone.

Step-3. We take VARL threshold value of a metric of JEdit-4.0 or Ant-1.3 as shown in Table 2. For example, the VARL value of WMC of JEdit-4.0 is 25.14.

Step-4. We select a subsequent release of JEdit-4.0 or Ant-1.3.

Step-5. For each class of selected systems, we use its metric value and corresponding VARL thresholds, and label the class as fault prone if metric value >= VARL otherwise not fault-prone.

Step-6. We create a confusion matrix which includes True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN). A class will be counted as True Positive (TP) if it is a faulty class in reality and through the threshold, it is also predicted as a faulty class. Similarly, a class will be counted as True Negative (TN) if it is not faulty in the reality and through the threshold; it is also predicted as a non faulty class. Subsequently, a class will be counted as False Positive (FP) if it is fault class in reality and through the threshold it is not predicted as a faulty class. Finally, a class will be counted as False Negative (FN) if it is not faulty in the reality and through thresholds it is predicted as a faulty class.

Step-7. We use Recall (equation 5), Precision (equation 6), F-measure (equation 7) and classification accuracy (equation 8) performance measures to evaluate the effectiveness of the derived VARL threshold value of considered metric such as WMC.

$$\text{Recall} = TP / (TP + FN) \tag{5}$$

$$\text{Precision} = TP / (TP + FP) \tag{6}$$

$$\text{F-measure} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{7}$$

$$\text{Accuracy} = (TP + TN) / (TP + FP + FN + TN) \tag{8}$$

Step-8. We repeat steps from Step-4 to Step-7 for each subsequent release of JEdit-4.0 and Ant-1.3 and corresponding VARL thresholds. The results of the evaluation process are shown in Table 3.

Table 3. Evaluation Results of Jedit-4.0 and Ant-1.3 Metrics VARL Thresholds

Metrics	JEdit-4.0 Subsequent Releases (Accuracy in %)			Ant-1.3 Subsequent Releases (Accuracy in %)			
	JEdit-4.1	JEdit-4.2	JEdit-4.3	Ant-1.4	Ant-1.5	Ant-1.6	Ant-1.7
WMC	78	86	91	74	88	76	80
DIT	67	73	79	73	82	67	72
NOC	72	85	96	76	88	73	77
CBO	74	80	79	74	86	74	77
RFC	81	80	74	70	87	81	82
LCOM	77	88	94	76	88	75	79
CA	74	82	84	75	87	74	77
CE	75	74	71	74	35	49	40
NPM	78	84	84	76	87	74	80
DAM	58	57	56	76	89	74	78
MOA	76	77	78	75	87	76	77
MFA	57	75	73	65	72	60	62
CAM	66	77	86	67	69	56	62

E. Third Analysis

In the third analysis, we performed the second experiment in order to validate the derived metric thresholds of our proposed model. We perform this experiment to provide a comparison of VARL thresholds with data distribution based threshold introduce by Erni and Lewerentz [15] and K.A.M. Ferreira's work [8]. Erni and Lewerentz identify a threshold value T of a metric via mean (μ) and standard deviation (σ) parameters with respect to data distribution of a corresponding metric that is T=μ+σ. The value of a metric greater than T refers to a problem. Similarly, in their study, Ferreira et al perform a system size analysis to identify the threshold values of five (out of six in their study) software metrics LCOM, DIT, Afferent Coupling (Ca), Number of Public fields and Number of Public Method (NPM). Moreover, they make three groups of the studied systems with respect to system size (# of classes) such as 1) the first group includes projects with system size less than 100 classes, 2) the second group includes

Authorized licensed use limited to: CITY UNIV OF HONG KONG. Downloaded on September 17,2026 at 02:35:39 UTC from IEEE Xplore. Restrictions apply.

Table 4. Comparison of Metrics Thresholds

Metrics	Accuracy of VARL (%)				Accuracy of Ferriera's thresholds (%)				Accuracy of μ + σ Thresholds (%)			
	Ant-1.4	Ant-1.5	Ant-1.6	Ant-1.7	Ant-1.4	Ant-1.5	Ant-1.6	Ant-1.7	Ant-1.4	Ant-1.5	Ant-1.6	Ant-1.7
DIT	73	82	67	72	61	55	52	54	71	73	61	61
LCOM	76	88	75	79	62	73	74	72	75	88	75	75
CA	75	87	74	77	71	85	73	76	72	85	74	74
NPM	76	90	74	80	76	88	74	78	74	85	76	76

projects with system size between 101 and 1000 classes, and 3) the third group of the project with system size greater than 1000 classes. Subsequently, Ferreira et al' uses three threshold values of the studied metrics (except DIT) and rank these three values as good, regular and bad. The bad rank or typical threshold values of LOCM (25), DIT (2), Ca (20) and NPM (30) for the first group and bad rank or typical threshold values of LOCM (20), DIT (2), Ca (20) and NPM (50) for the second group of projects. In our second experiment, we follow the same steps which are described

Table 5. Association of CBO and DIT Thresholds

Projects	Accuracy of VARL (%)	Accuracy of μ + σ (%)
Ant-1.4	76	76
Ant-1.5	89	88
Ant-1.6	74	73
Ant-1.7	78	77
Jedit-4.1	77	75
Jedit-4.2	87	89
Jedit-4.3	95	94

Table 6. VARL thresholds for 31 projects

Project	WMC	DIT	NOC	CBO	RFC	LCOM	CA	CE	NPM	DAM	MOA	MFA	CAM
Berek	004	01.4	0.2	05.5	29.3	183	2.3	01.4	21.7	0.6	-00.4	-2.5	0.5
e-learning	11.7	4.2	-0.1	08.8	22.9	63.0	3.5	04.5	11.3	0.9	00.8	5.4	0.3
Forrest-0.8	10.5	1.3	-0.3	16.4	46.4	42.5	0.1	15.9	8.1	1.5	11.3	-0.2	0.3
Intercafe	19.3	22.8	0.2	09.5	38.5	-11.9	6.3	03.8	2.7	-0.1	-0.2	0.9	0.4
Kalkulator	8.9	2.4	0.3	14.7	-1213	32.6	2.2	18.4	5.8	-0.1	0.3	0.5	0.2
Nieruchomosci	-06.7	-00.5	0	-3.2	6.4	-13.9	-37.2	-2.9	-11.8	-0.9	-2.5	-1.6	-2.5
Pbeans2	04.9	3.1	-0.8	2.1	-3.6	-49.4	0.1	-0.4	4.1	0.1	-1.1	0.7	0.7
Pdftranslator	0.7	-01.1	1710	0.9	1.0	-03.2	-2.2	-0.8	1.2	3.9	-0.7	-1.0	0.9
Serapion	06.1	-0.8	-20.1	1.1	19.2	07.8	-6.6	0.3	3.5	-0.3	0	-2.1	0.8
Skarbonka	0.4	0.4	0	-11.8	8.3	-64.3	-56.1	-5.7	-1.9	0.2	-2.3	-0.5	0.7
Szybkafucha	12.5	0.9	0	-2.1	-40.4	33.6	8.3	-1	12	-223	-1.1	-0.1	1.2
termproject	02.1	-00.5	0.5	-1.5	4.2	-06.5	-8.3	-1.6	-6.7	-1.1	-10.1	-1.1	0.7
workflow	-19.6	26.0	18.1	42.7	-33.9	-239.9	12.9	-8.5	-15.9	-8.6	-2.4	9.9	1.4
wspomaganiepi	2.5	-04.6	0.3	1.9	4.2	-7.4	0.1	-1.0	-2.5	-1.7	-1.06	-1.5	0.8
zuzel	0.2	-0.7	0	-39.5	12.3	-31.9	-128	-0.4	33.4	-2.1	-0.8	-0.5	0.8
sklebagd	-13.9	3.9	0.3	-00.6	06.9	-668.9	-08.5	-3.1	-15.4	-0.8	-18.5	1.8	0.9
systemdata	07.8	0.7	0	06.8	17.5	20.3	3.9	3.0	0.5	0.1	-1.9	-1.8	0.5
Ant-1.7	2.6	-12.2	53.9	-63.8	15.5	-127.9	-256	01.4	0.6	-0.4	-1.3	9.8	0.6
Camel-1.6	-11.3	-11.4	-05.3	-25.1	-26.0	-930.4	-34.1	-25.1	-13.2	-0.9	-3.8	6.0	1.2
Ivy-2.0	11.7	06.3	1.26	12.1	41.4	175.5	02.1	05.3	7.9	0.6	0.5	0.5	0.4
JEdit-4.3	168	14.9	-03.5	137	261.9	28492.6	121.9	40.9	86.1	49.5	7.9	25.0	-0.9
Log4j-1.2	-40.5	73.5	-00.1	-17.6	-265.5	-434.2	-08.8	-66.6	-19.2	-1.8	-178.7	8.4	4.9
lucene-2.4	-15.2	-08.8	-13.2	-17.0	-34.5	-163.4	-45.3	-13	-10.5	-1.6	-22.2	-14.8	1.7
Poi-3.0	-11.9	16.5	-869	-41.6	-27.0	-193.6	-403	-07.1	-15.9	-1.4	-5.1	770.4	-3.8
Prop-6	10.5	01.3	0	11.9	29.7	58.7	05.1	07.3	8.7	0.6	1.2	0.2	0.4
redaktor	28.1	01.2	00.1	-142	148.6	0.2	17.9	-0.41	31.0	1.3	8.8	0.2	0.3
tomcat	20.9	00.7	02.4	12.6	55.7	539.7	09.1	0	18.4	0.8	1.8	-0.4	0.4
velocity-1.6	-13.5	05.5	-09.7	-28.4	-26.6	345	182.2	-02.8	12.3	-2.4	-2.4	2.8	0.7
xalan-2.7	-27.9	-13.3	863	-28.7	-46.2	-61.9	-63.0	-17.5	-2.3	-3.5	-25.1	-6.4	4.7
xerces-1.4	-119	-2.0	-09.5	-02.9	-57.9	-3797	-09.1	-0.9	-2213	-1.9	-6.4	-1.1	1.9
synapse-1.2	-19.5	-07.5	-242	-11.5	-04.9	-1796	-68.5	-3.5	-33.4	-0.6	-2.1	-5.3	0.9

in Section IV-D, however, we replace the reference of VARL threshold (step-3 and step-5 in Section IV-D) with Erni and K.A.M. Ferreira's metrics thresholds. Moreover, we only use the subsequent releases of the Ant-1.3 project to compare the effectiveness of metrics thresholds in order to classify the fault-prone classes. The results of the experiment are shown in Table 4. Subsequently, the correlation of object-oriented metrics can also play an additional role to explain the defect complexity and elaborate the software quality, such as the interaction of CBO and DIT can help to explain the more defect [23]. Moreover, we performed our third experiment to evaluate and compare the effectiveness of VARL thresholds with Erni's thresholds $T=\mu+\sigma$ in the existence of a correlation between metrics. In this experiment, we follow the same steps as described in Section IV-D and use subsequent releases of Ant-1.3 and JEdit-4.0 projects, however, we consider the interaction of CBO and DIT metrics

Authorized licensed use limited to: CITY UNIV OF HONG KONG. Downloaded on September 17,2026 at 02:35:39 UTC from IEEE Xplore. Restrictions apply.

thresholds to classify the fault-prone classes. The results of this experiment are shown in Table 5.

F. Fourth Analysis

We perform this analysis in order to investigate the applicability of the proposed model for the all projects used in this study, and consider it as a benchmark to derive the metrics thresholds. We perform two tasks in this analysis. In the first task, we collect faulty data of 31 open source projects from PROMISE repository. The system size (# of classes) and the number of classes associated with bugs are varied, such as system size and a number of classes (associated with at least one bug) of Velocity-1.6 are 229 and 34 respectively. In the second task, we used the proposed model and compute the VARL metrics thresholds for each project. We follow the procedure described in Section III and compute the VARL (Value of An Acceptable Risk Level) of each metric (defined in Section IV-B). The results of the analysis are shown in Table 6.

V. RESULTS AND DISCUSSION

We have summarized the results of the four analyses in order to present the effectiveness of derived thresholds (that is VARL) and the applicability of the proposed model in the domain of classification of fault proneness classes.

- The results of Table 2 and Table 3 suggest the significance of the proposed model to derive thresholds of studied metrics (section IV-B). Moreover, these results also suggest the effectiveness of derived thresholds which is assessed through their classification accuracies in subsequent releases of JEdit-4.0 and Ant-1.3 open source projects as shown in Table 3. For example, the effectiveness of derived threshold for WMC (that is VARL = 25.14 for JEdit-4.0) can be observed through its classification accuracies 78%, 86% and 91% in subsequent releases JEdit-4.1, JEdit-4.2, and JEdit-4.3 respectively.
- The interesting implication of the results (shown in Table 2 and Table 3) is the support for the applicability of the proposed model to derive thresholds for the all projects. These results aid to increase the belief of a specialist to use the proposed model to derive more precise thresholds for other metrics, which follow power law distribution.
- The comparative results of Table 4 suggest that the proposed model may be applicable to derive thresholds (typical value) for those metrics which follow the heavy-tail distribution and need to define their reference values. Such as in their study, K.A.M. Ferreira [8] reported the reference values for the Number of Public Method (NPM), LCOM and Ca metrics. The effectiveness of typical values (VARL thresholds) of these metrics can be observed from comparative results (shown in Table 4). For example, the effectiveness (in term of classification accuracy) of VARL threshold of these metrics can be observed significantly better than the bad reference threshold [8] values of these metrics.
- Moreover, from the results of Table 4, we can also observe the effectiveness of VARL thresholds significantly better than the data distribution based thresholds [15]. However, in the case of metric correlation such as CBO and DIT [23], the comparative results of VARL and data distribution based threshold $T=\mu+\sigma$ are shown in Table 5. We cannot observe a significant increase in the effectiveness of VARL thresholds as compared to K. Erni thresholds. Consequently, in order to present a significant difference between the effectiveness of thresholds, we apply two-sided Wilcoxon Signed-rank test at $\alpha=0.05$ and observe the significant difference in the effectiveness of both thresholds.

Table 7. VARL Significance and Range

Metric	# of Projects (VARL Signficant)	# of Projects (VARL not Significant)	VARL range for project used in study
WMC	20	11	0.20 to 168
DIT	18	13	0.40 to 73.50
NOC	14	17	0.20 to 1710
CBO	15	16	0.90 to 137.00
RFC	18	13	1.00 to 261.90
LCOM	12	19	0.20 to 28492
CA	14	17	0.10 to 182.20
CE	12	19	0.30 to 40.90
NPM	17	14	0.60 to 86.10
DAM	11	20	0.10 to 49.50
MOA	08	23	0.30 to 11.30
MFA	15	16	0.20 to 25.90
CAM	28	03	0.30 to 04.90

- The result of Table 6 suggests that the proposed model is not applicable to derive significant VARL thresholds for the all projects due to the existence of negative values, which are considered as out of range [18] and cannot be used in the evaluation.
- The result of Table 6 suggests that like other metrics threshold techniques, proposed model is also related to data characteristics such as distribution (normal or heavy-tailed) of a metric data. From the results of Table 6, we can also observe that proposed model cannot identify a particular metric threshold that can fit for all projects and assure the relevant quality attribute.
- However, through the results of Table 6, we can infer the significance of the proposed model through 1) to count the average number of projects for which significant VARL threshold for the corresponding metric is derived, such as we can observe significant VARL threshold for CAM metric in 28 out of 31 projects (shown in Table 7), and 2) the recommended range of VARL thresholds for corresponding metrics. We have summarized and presented these results in Table 7.

VI. THREAT TO VALIDITY

In this study, we also found some threat to validity. The first threat to validity is relevant to the collection of faulty data and use of metrics which have more than a definition or variants such as WMC has two definitions. The faulty data on study projects might be extracted using tools which have different definitions or variants with a different definition of metrics. The second threat is related to the selection of a value for P_0 which is 0.1 in our study in order to derive the value of VARL for a

Authorized licensed use limited to: CITY UNIV OF HONG KONG. Downloaded on September 17,2026 at 02:35:39 UTC from IEEE Xplore. Restrictions apply.

metric. The P_0 parameter in VARL calculation is subjective, consequently, this study can be replicated by adjusting other values of P_0 and calculate the more significant metrics thresholds.

VII. Conclusion

Software metrics can serve many purposes for developers and stakeholders throughout the software development life cycle. Subsequently, a developer can assure the quality of software products through interpretation of metric values. In the domain of epidemiology study, R. Bender introduced an approach to derive thresholds to quantify the risks associated with risk factors. In this study, we follow the Bender's approach and proposed a model to derive thresholds for a set of object-oriented metrics. We performed four types of analysis and three experiments to derive and evaluate the effectiveness of derived thresholds in order to detect and classify the fault-prone classes. We retrieved 31 open source projects and subsequent releases of JEdit-4.0 and Ant-1.3 from PROMISE repository and used confusion matrix based performance measures to evaluate the metrics threshold effect in the domain of classification of fault-prone classes. In an experiment, we compare and validate the effectiveness of derived metrics thresholds with data distribution based thresholds of DIT, LCOM, Ca and NPM. The analysis and experiments results suggest the applicability of the proposed model in order to derive thresholds of metrics either they follow the heavy-tailed distribution or not. Moreover, in comparison of derived thresholds (VARL) with data distribution based thresholds, we found a significant increase in classification accuracy of VARL thresholds. For example, In comparing the effectiveness of VARL thresholds with data distribution thresholds of DIT, CA, LCOM, and NPM, we observed a significant increase in classification accuracy of DIT (27%), Ca (2%), NPM (2%) and LCOM (15%) for the Ant-1.5 project. Finally, we concluded that the proposed model is not applicable to derive significant metrics thresholds for the all studied systems. However, all significant VARL thresholds remain effective in the classification of fault-prone classes, which can aid developers to make decisions in order to improve software quality.

VIII. References

[1] J. Bansiya, and C.G.A. Davis., A hierarchical model for object-oriented design quality assessment.IEEE Transactions on Software Engineering, 28(1): p. 4-17, 2002.

[2] R. Linckeet al., Comparing software metrics tools, in Proceedings of the 2008 international symposium on Software testing and analysis, Seattle, WA, USA. p. 131-142,2008.

[3] S. R. Chidamber and C. F. A. Kemerer., metrics suite for object oriented design.IEEE Transactions on Software Engineering, 20(6): p. 476-493,1994.

[4] F. B. Abreu and R. Carapuça., Object-Oriented Software Engineering:Measuring and Controlling the Development Process, in International Conference on Software Quality. 1994.

[5] M. Shepperd, D. Bowes and T. Hall., Researcher Bias: The Use of Machine Learning in Software Defect Prediction.IEEE Transactions on Software Engineering, 40(6): p. 603-616, 2014.

[6] Dr. Radjenović et al., Software fault prediction metrics: A systematic literature review. Information and Software Technology, 55(8): p. 1397-1418, 2013.

[7] L. H. Rosenberg, Applying and Interpreting Object Oriented Metrics in Software Technology Conference. Utah, 1998.

[8] K. A. M. Ferreira et al., Identifying thresholds for object-oriented software metrics. Journal of Systems and Software, 85(2): p. 244-257, 2012.

[9] R. Shatnawi, A Quantitative Investigation of the Acceptable Risk Levels of Object-Oriented Metrics in Open-Source Systems, IEEE Transactions on Software Engineering, 36(2): p. 216-225, 2010.

[10] R. Shatnawi, Deriving metrics thresholds using log transformation. Journal of Software: Evolution and Process, 27(2): p. 95-113, 2015.

[11] P. D. S. Benlarbi et al. Thresholds for object-oriented measures, Proceedings of 11th International Symposium of Software Reliability Engineering, 2000. Proceedings. 11th

[12] K. E. Emam et al., The Optimal Class Size for Object-Oriented Software. IEEE Transactions on Software Engineering, 28(5): p. 494-509, 2002.

[13] R. Martin, OO Design Quality Metrics - An Analysis of Dependencies, Proceeding of Workshop on Pragmatic and Theoretical Directions in Object-Oriented Software Metrics. 1994.

[14] M. Jureczko and L. Madeyski, Towards identifying software project clusters with regard to defect prediction, in Proceedings of the 6th International Conference on Predictive Models in Software Engineering., ACM, Romania. p. 1-10, 2010.

[15] K. Erni and C. Lewerentz, Applying design-metrics to object-oriented frameworks, Proceedings of the 3rd International Symposium in Software Metrics, 1996.

[16] French, V.A., Establishing software metric thresholds, in International Workshop on Software Measurement. 1999.

[17] S. Hussain., Threshold analysis of design metrics to detect design flaws: student research abstract, Proceedings of the 31st Annual ACM Symposium on Applied Computing (SAC'16), p. 1584-1585, 2016.

[18] R. Bender, Quantitative Risk Assessment in Epidemiological Studies Investigating Threshold Effects. Biometrical Journal, 41(3): p. 305-319, 1999.

[19] K. Ulm, A statistical method for assessing a threshold in epidemiological studies. Statistics in Medicine, 10(3): p. 341-349, 1991.

[20] J. D. W. Hosmer, S. Lemeshow And R. X. Sturdivant, , Introduction to the Logistic Regression Model, Applied Logistic Regression., John Wiley & Sons, Inc. p. 1-33, 2013.

[21] M. Fowler et al., Refactoring: Improving The Design Of Existing Code., Berkeley California: Addison Wesley. 337, 2002.

[22] R. Baeza-Yates, and B. Ribeiro-Neto, Modern Information Retrieval, MA, USA: Addison-Wesley, 1999.

[21] R. Subramanyam and M.S. Kishnan, Empirical Analysis of CK Metrics for Object-oriented Design Complexity : Implication for Software Defects, IEEE Transaction on Software Engineering, 29 (4), p. 297-310, 2003.

Authorized licensed use limited to: CITY UNIV OF HONG KONG. Downloaded on September 17,2026 at 02:35:39 UTC from IEEE Xplore. Restrictions apply.