IOSR Journal of Engineering (IOSRJEN)
ISSN: 2250-3021 Volume 2, Issue 8 (August 2012), PP 190-195
www.iosrjen.org

Risk chain prediction metrics for predicting fault proneness in Software Systems

[1]N.Rajasekhar Reddy [2] Rang swami
[1] *Associate professor, Department of CSE, Madanapalli Institute of Technology and Science, Madanapalli, Andhra Pradesh, India.*
[2] *Research Scholar, Department of CSE, Madanapalli Institute of Technology and Science,Madanapalli,Andhra Pradesh, India.*

Abstract: *- The paper presents two atypical risk chain prediction metrics for barometer coupling and accord in software systems. Our aboriginal metric, Ideal Coupling between Object classes (ICBO), is based on the acclaimed CBO coupling metric, while the added metric, Ideal Lack of Cohesion on Methods (ILCOM5), is based on the LCOM5 accord metric. One advantage of the proposed risk chain prediction metrics is that they can be computed in a simpler way as compared to some of the structural metrics. We empirically advised ICBO and ILCOM5 for admiration fault proneness of classes in a ample accessible antecedent arrangement and compared these metrics with a host of absolute structural and risk chain prediction metrics for the aforementioned task.*

I. Introduction

Coupling and cohesion measures capture the degree of interaction and relationships among source code elements, such as classes, methods, and attributes in object oriented (OO) software systems. One of the main goals behind OO analysis and design is to implement software system where classes have high cohesion and low coupling among them. These class properties facilitate comprehension activities, testing efforts, reuse, and maintenance tasks.

A vast majority of coupling and cohesion metrics abound in the literature relies on structural information, which captures relations, such as method calls or attributes usages. These metrics have been proved useful in different tasks, such as, assessment of design quality[4, 10], impact analysis [8, 36, 41], prediction of software quality [26], and faults [17, 22, 37],identification of design patterns [2] etc. However, these structural metrics lack the ability to identify ideal links, which, for example, specify implicit relationships encoded in identifiers and comments in source code.

In this paper we propose two new risk chain prediction metrics, namely Ideal Coupling between Object classes (ICBO) and Ideal Lack of Cohesion of Methods (ILCOM5) metrics. The proposed metrics are different from existing ideal coupling metrics [35]and cohesion [31] metrics as they utilize different counting mechanisms inspired by peer structural cohesion and coupling metrics.

In order to evaluate the proposed metrics, we compare ICBO and ILCOM5 against a large host of existing structural and ideal coupling metrics for predicting faults in a large open source software system. Furthermore, we perform a comprehensive empirical evaluation of other parameters, such as, impact of preprocessing techniques. Such parameters also impact performance of other existing risk chain prediction metrics, such as Ideal Cohesion of Classes (C3) [31] and Ideal Coupling among Classes (CoCC) [35]. The results of our empirical study indicate that ICBO andILCOM5 not only can be used to build operational models for predicting fault proneness of classes, but can also be effectively used in conjunction with other structural metrics to improve overall accuracy of bug prediction models.

Our paper warrants the following contributions:

- We define two new ideal cohesion and coupling metrics, which are easier to compute than their structural protégé.
- We carried out an extensive empirical study of 61software metrics, including newly proposed measures to build models for fault prediction using machine learning and logistic regression analyses.
- We empirically studied a range of parameters that can impact performance of ICBO and ILCOM5metrics, such as, impact of corpus stemming and parameterized thresholds.
- We developed an online appendix summarizing the results of our empirical study to facilitate development and comparison of risk chain prediction metrics and ensure reproducibility of our results.

II. Risk chain prediction metrics

Our approach to measuring coupling and cohesion relies on the assumption that the methods and classes of Object Oriented systems are connected in more than one way. While the most explored and evaluated set of relations among methods and classes are based on data and control dependencies, in this work we rely on orthogonal type of relationships, known as ideal dependencies to capture ideal cohesion and coupling of classes.

Ideal coupling and cohesion metrics, such as, CoCC and C3 extract, encode, and analyze the semantic information embedded in the comments and identifiers in software. Software developers utilize the comments and identifiers to represent elements of the problem or solution domain [11, 15]. Whilst ideal cohesion[31] and coupling [35] metrics capture this information and have been proposed elsewhere in the research literature, we augment a family of risk chain prediction metrics with two new members, namely ICBO and CLOM5.Our metrics rely on the equivalent underlying mechanism to extract and analyze the ideal information from the identifiers and comments in source code as previous risk chain prediction metrics, which are based on Latent Semantic Indexing (LSI) [14]. LSI has been used before to support other source code analysis tasks such as concept location [34], identification of abstract data types [29], clone detection [40], traceability link recovery among software artifacts [1, 13, 30], software clustering [25], quality assessment [26] and software measurement [16, 31, 32, 35, 36]. For the sake of completeness we provide some of the details on the LSI in the next section.

1.1 Concealed Semantic Indexing in the Nutshell

LSI is a machine-learning model that induces representations of the meaning of words by analyzing the relations among words and documents in textual corpus of data. LSI was initially developed in the context of information retrieval as a way of overcoming issues with polysemy and synonymy, which are inherent to the vector space model (VSM) [38]. The specific technique, which is used by LSI to capture vital ideal information and tackle two aforementioned problems, is dimension reduction, which implies selecting the top dimensions from a co-occurrence term document matrix decomposed using singular value decomposition (SVD). Consequently, LSI provides an effective mechanism to assess and evaluate similarity amid any two documents in the text corpus (i.e., methods in software) in an unsupervised fashion. While the details behind SVD are rather complex and lengthy to be presented in this paper, we refer a reader to [14].LSI relies on VSM, which is an extensively used approach for encoding documents in the corpus as numerical vectors. More specifically, VSM encodes a corpus by a term-by-document matrix whose $[i, j]^{th}$ element indicates the association between the i^{th} term and j^{th} document. In case of our particular application, a term is an identifier or a comment, and a document is abode of the method extracted from a source code file. The foundation of VSM lies in the mechanism that represents documents by its association with terms where the association is measured by term cooccurrencesin the documents. There are several mechanisms to capture these associations based on term occurrences such as term frequencies (default case in our empirical evaluation) and term frequency – inverse document frequency (tf-idf). In a term-by-document matrix, a tf-idf value for $[i, j]^{th}$ element implies a statistical measure evaluating how important a word into a document in a corpus. Formally,

$$w_{t,d} = tf_{t,d}.\log N / df_t$$

Here $tf_{t,d}$ is the term frequency of a document d, and df_t is the term frequency in all the documents in the corpus, whereas the N is the number of documents in the corpus. The importance of a word increases proportionally to the number of times a word appears in the document, but is offset by the number of times of that word appearing in the corpus [38].

The ideal similarity between documents is measured via the cosine or inner product between the corresponding vectors (i.e., methods), which increases if more words are shared. This underlying mechanism entirely supports the idea of measuring ideal coupling and cohesion in software based on word matching from identifiers and comments in software.

1.2 Ideal Cohesion & Coupling Metrics

The definitions of the new ideal cohesion and coupling of classes builds on our previous work for measuring the ideal cohesion [32] and coupling[36] of classes. The source code of the software system is parsed and transformed into a corpus of textual documents where each document corresponds to the implementation of a method. Aforementioned LSI technique takes the corpus as an input and creates a term-by-document matrix, which captures the dispersion and co-occurrence of terms in class methods. SVD issued next to construct a subspace, referred to as the LSI subspace. All methods from this matrix are represented as vectors in the LSI subspace. The cosine similarity between two vectors is used as a measure of ideal similarity between two methods and is purported to determine shared ideal information between two methods in the context of the entire software system. This mechanism to capture ideal similarity among documents has been introduced before in Ideal Coupling of Classes and Ideal Cohesion of Classes measures and is also used here. Next we define the model, ICBO, and CLOM5measures. Some of the definitions have been presented elsewhere [35], however, we also include them for the sake of completeness.

1.3 Principal Definitions

Definition 1 (System, Classes, Methods). We define an OO system as a set of classes C = $\{c_1, c_2 ... c_n\}$ with the number of classes in the system n = |C|. A class has a set of methods. For each class c ∈ C, M(c) = $\{m_1, m_2 ... m_t\}$ represent its set of methods, where t = |M(c)| is the number of methods in a class c. The set of all the methods in the system is denoted as M(C).

An OO system C can be also viewed as a set of connected graphs G_c = $\{G_1, ... G_n\}$ with G_i representing class C_i. Each class C_i ∈ C is also represented by a graph G_i ∈ G_c such that G_i = (V_i, E_i), where V_i = M(C_i) is a set of vertices corresponding to the methods in class C_i and E_i ⊂ V_i * V_i is a set of weighted edges that connect pairs of methods from the class.

Definition 2 (Ideal Similarity between Methods). The ideal similarity between methods(CSM) $m_k \in M(C)$ and $m_j \in M(C)$, CSM(m_k, m_j), is computed as the cosine amid two vectors vm_k and vm_j,representing m_k and m_j in the LSI semantic space:

CSM(m_k, m_j) $= = \dfrac{vm^t{}_k vm_j}{|vm_k|_2 \times |vm_j|_2}$

As defined, the value of CSM(m_k, m_j)∈ [-1, 1], as CSM is a cosine similarity in the LSI space. In order to fulfill non-negativity property of software metrics [9], we refine CSM as the following:

$$CSM^1(m_k, m_j) = \{CSM(m_k, m_j) if CSM(m_k, m_j) > 0$$

Else 0

CSM^1 has been used as a base for defining C3 [31]and CoCC [35] measures before.

Definition 3 (Parameterized Ideal Similarity)In our work we define ideal cohesion and coupling metrics utilizing counting mechanisms, stemming from existing structural metrics, which are sensitive to the input information such as nodes and edges (e.g., methods and attribute references). Thus, in this work we introduce a notion of a parameterized ideal similarity, which distinguishes among significant and non-significant ideal interactions among methods of classes. In particular, we conjecture that it is possible to empirically derive a threshold for a given software system to distinguish between strong and weak ideal similarities. More formally, we define parameterized CSM^p as:

$$CSM^p(m_k, m_j, t) = \{1 if CSM^1(m_k, m_j) > t$$

Else 0

Of course, the particular threshold t depends on the specific software system. In our previous experience, the absolute value of the cosine similarity can not be used as a reliable indicator of presence or absence of ideal relationship among pairs of methods as more comprehensive analysis of similarity distributions is required. One of the main research questions in our empirical evaluation is centered on empirically deriving such a threshold and analysis of the impact on the choice of threshold values on the resulting metrics.

1.4 Ideal Lack of Cohesion in Classes

In this paper we define our first metric, namelyILCOM5 using CSMP as the foundation for computing ideal similarities among methods of classes, however, in terms of counting mechanism we rely on one of the ideas from previously defined structural metrics, namely LCOM5 [23], graph based cohesion metric. The main difference between our metric,ILCOM5 and C3, ideal cohesion of classes metric, is that we define a parameterized version of cohesion metric using a different counting mechanism:

CLCOM 5(c, x) = NoCC(G),

where NoCC identifies the number of connected components in the graph $G_C = (M(c), E), c \in C, E \subseteq M(c) \times M(c)$, and ($m_k$, m_j) ∈ E if CSMP (m_k, m_j, t)=1.

1.5 Ideal Coupling between Object classes

The definition of ICBO relies on previous definitions for CoCC metric. We provide these definitions and explain how we adjusted them in the current work.

Let $c_k \in C$ and $c_j \in C$ be two distinct $(c_k \neq c_j)$ classes in the system. Each class has a set of methods $M(c_k) = \{mk_1, ...mk_r\}$, where r = $|M(c_k)|$ and $M(c_j) = \{mj_1, ...mj_t\}$, where t = $|M(c_j)|$. Between every pair of methods (m_k, m_j) there is a similarity measure CSMP(m_k, m_j). We can similarly define the ideal similarity between two classes c_j and c_k, that is CSCP , as follows:

$$CSC^p\ (c_k,\ c_j,\ \mathrm{t}) = \{1 if CSC^1(C_k, C_j) > t$$

The definition ensures that the ideal similarity between two classes is symmetrical, as $CSC\ (c_k, c_j) = CSC(c_j, c_k)$. In this case we use class granularity to build the corpus. This is the main difference between computing ILCOM5 and ICBO metrics. We refine the ideal similarity for a class c as the following:

$$\mathrm{ICBO(c, t)} = \sum_{c_k \in C, c \neq c_k} CSC^p(c, c_k, t)$$

which is the sum of the parameterized ideal similarities between a class c and all the other classes in the system.

III. Evaluations of Metrics and Analysis

Precision, recall and accuracy are three widely used information retrieval metrics that were employed to measure performance of software metrics for various maintenance tasks including predicting fault-prone classes. We explain these measures in the context of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN) in the context of fault proneness prediction. True Positive is a candidate class, which was predicted as faulty and contained a bug, whereas a True Negative is a candidate class predicted as non-faulty and containing no bugs. The False Positive is class that was marked as faulty by the model, but actually did not contain a fault, whereas a False Negative is a class where the model marked a class as non-faulty and it did contain a bug. Accuracy, Precision and Recall are defined as the following:

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN}, \Pr ec = \frac{TP}{TP+FP}, \mathrm{Re}\, call = \frac{TP}{TP+FN}$$

While reporting the results we also used F-measure, which is a harmonic mean of precision and recall. In order to explore the principal, orthogonal dimensions captured by the coupling and cohesion measures (both ideal and structural) we performed Principal Component Analysis (PCA) on the metrics. Applying PCA to metrics data consists of the following steps: collecting the metrics data, identifying outliers, and performing PCA. We applied PCA in the similar manner as in our previous work [31, 35], including procedures for identifying outliers and rotating principal components. In general, via PCA we can recognize groups of variables (i.e., metrics), which are likely to measure the same underlying dimension (i.e., specific mechanism that defines, for instance, coupling or cohesion) of the object to be measured (e.g., cohesion of a classes).

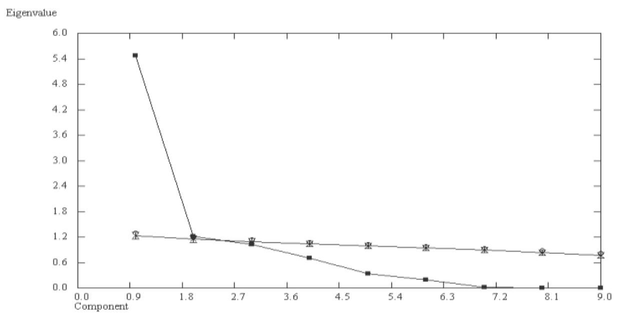
IV. Related Work

Our related work can be broadly classified into two areas – ideal cohesion and coupling metrics and predicting fault-proneness of classes. Ideal cohesion of classes or C3 [31] is one of the first risk chain prediction metrics proposed in the research literature. C3, similarly to ILCOM5 and ICBO is based on the analysis of the semantic information embedded in the source code, such as identifiers and comments. C3 has been recently used in conjunction with structural cohesion metrics to predict faults in object-oriented classes [32]. CoCC [35] is a ideal coupling metric, also based on LSI, stems from C3,however, it was defined to capture coupling among classes based on ideal similarities among methods in different classes. CoCC has been shown to outperform structural coupling metrics for the task of impact analysis on a large open-source system [36].Finally, WME is a ideal cohesion metric based on Latent Dirichlet Allocation and information theory approaches [28]. This cohesion metric has been shown to capture different aspects of class cohesion and improved fault prediction for most existing cohesion metrics. While building comprehensive models for fault prediction was not at the focus of papers presenting risk chain prediction metrics, this paper not only introduces new metrics, but also explores their role in building complete models for fault prediction.

Existing research showed that software metrics can be used as good indicators for the fault proneness of classes in OO systems [3, 5, 7, 10, 17, 22, 33, 37, 39].More specifically, some of the existing approaches also utilized machine learning [22] and logistic regression analyses [3, 5, 7, 10, 22, 33, 39] to build metric-based models for fault prediction. Our paper is different from the previous work as it defines new risk chain prediction metrics for class cohesion and coupling, which appear to be an improvement over the state-of-the-art. Finally, this work explores a set of machine learning techniques and regression analyses to test a number of models based on the combinations of structural and risk chain prediction metrics along with the detailed investigation into principal factors impacting the performance of the risk chain prediction metrics. Finally, prediction of fault-prone classes or simply bug prediction is an active area of research, which produced a number of research publications in the last decade. Besides conference and journal publications on the topic, specialized conferences were organized such as PROMISE6 and MSR7 with their specialized data sets for predicting fault-prone classes in software. The performance of the proposed metrics was test on various open source java application builds

The sensitivity of the fault proneness prediction can measured by applying principal component analysis [see the fig 1] . Comparison of fault proneness sensitivity between CBO&LCOM5 and ICBO&ILCOM5 can be found in fig 2. The fault prediction sensitivity measure as

$$\text{Sensitivity}=\frac{\text{Classes correctly predicted as fault prone}}{\text{Classes actually fault prone}}$$



PCA analysis of the CBO&LCOM5 and ICBO&ILCOM5

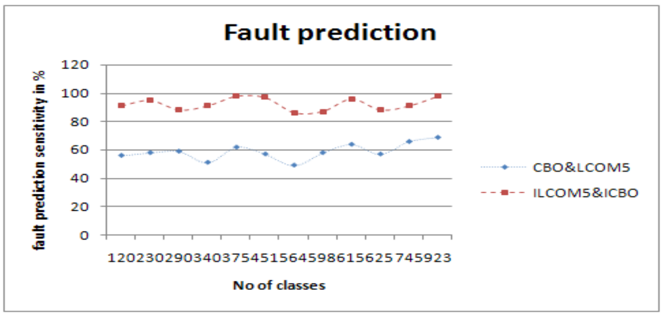


Fig 2: Fault prediction sensitivity comparison between CBO&LCM5 and ICBO&ILCOM5

V. Conclusion

The paper defines novel operational measures for ideal class cohesion and coupling measurement, which have been empirically validated. An extensive case study using machine learning techniques on metrics data indicates that the proposed measures have comparable accuracy with those defined suing structural information. Moreover, combinations of novel metrics with existing host of measures attests statistically significant improvement in the results across multiple evaluation criteria.

The paper lays the basis for the future work that makes use of ideal information for coupling and cohesion measurement. The proposed metrics could be further extended and refined, for instance, by taking into account inheritance. Another direction is to improve the quality of the underlying textual information by applying advanced source code pre-processing techniques for splitting [18] and expanding [24, 27]compound identifiers and comments in software. Since both ICBO and ILCOM5 rely on textual (unstructured) information, we are considering including external documentation in the corpus, which should extend the context in which words are used in software to capture underlying ideal similarities.

References

[1] Antoniol, G., Canfora, G., Casazza, G., De Lucia, A., and Merlo, E.,"Recovering Traceability Links between Code and Documentation", IEEE Transactions on Soft. Engineering, vol. 28, no. 10, pp. 970 - 983. 2002

[2] Antoniol, G., Fiutem, R., and Cristoforetti, L., "Using Metrics to Identify Design Patterns in Object-Oriented Software", in Proc. of 5th IEEEMETRICS'98, Bethesda, MD, pp. 23 - 34., 1998

[3] Arisholm, E., Briand, L. C., and Foyen, A., "Dynamic coupling measurement for OO software", IEEE TSE, vol. 30, no. 8, pp. 491-506., 2004

[4] Bansiya, J. and Davis, C. G., "A hierarchical model for object-oriented design quality assessment", IEEE TSE, vol. 28, no. 1, pp. 4-17., 2002

[5] Basili, V. R., Briand, L. C., and Melo, W. L., "A Validation of OO Design Metrics as Quality Indicators", IEEE TSE, vol. 22/10, Oct., pp. 751-761., 1996

[6] Basili, V. R., Caldiera, G., and Rombach. D. H., The Goal Question Metric Paradigm, John W & S, 1994.

[7] Briand, L., Melo, W., and Wust, J., "Assessing the Applicability of Fault-Proneness Models across OO Software Projects", TSE, vol.28/7,706-720., 2002

[8] Briand, L., Wust, J., and Louinis, H., "Using Coupling Measurement for Impact Analysis in OO Systems", in IEEE ICSM'99, pp. 475-482., 19996

[9] Briand, L. C., Daly, J., and Wüst, J., "A Unified Framework for Coupling Measurement in OO Systems", IEEE TSE, vol. 25/1, pp. 91-121., 1999

[10] Briand, L. C., Wüst, J., Daly, J. W., and Porter, V. D., "Exploring the relationship between design measures and software quality in object-oriented systems", Journal of System and Software, vol. 51, no. 3, pp. 245-273., 2000

[11] Caprile, B. and Tonella, P., "Restructuring Program Identifier Names", inProc. of 16th IEEE ICSM'00, San Jose, California, USA, pp. 97-107., 2000

[12] Chidamber, S. R. and Kemerer, C. F., "Towards a Metrics Suite for Object-oriented Design", in Proc. of OOPSLA'91, pp. 197-211., 1991

[13] De Lucia, A., Fasano, F., Oliveto, R., and Tortora, G., "Recovering Traceability Links in Software Artifact Management Systems", TOSEM, vol.16, no. 4, 2007.

[14] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R., "Indexing by Latent Semantic Analysis", Journal of the American Society for Information Science, vol. 41, pp. 391-407. , 1990

[15] Deissenboeck, F. and Pizka, M., "Concise and Consistent Naming", Software Quality Journal, vol. 14, no. 3, pp. 261-282, 2006

[16] Dit, B., Poshyvanyk, D., and Marcus, A., "Measuring the Semantic Similarity of Comments in Bug Reports", in Proc. of 1st STSM'08, 2008

[17] El-Emam, K. and Melo, K., "The Prediction of Faulty Classes Using Object-Oriented Design Metrics", NRC/ERB-1064, vol. NRC 43609, 1999.

[18] Enslen, E., Hill, E., Pollock, L., and Vijay-Shanker, K., "Mining Source Code to Automatically Split Identifiers for Software Analysis", in Proc. of 6thIEEE MSR, Vancouver, BC, Canada, pp. 71-80., 2009

[19] Ferenc, R., Beszedes, A., and Gyimóthy, T., "Extracting Facts with Columbus from C++ Code", in Proc. of 8th CSMR"04, March, pp. 4-8., 2004

[20] Ferenc, R., Beszédes, Á., Tarkiainen, M., and Gyimóthy, T., "Columbus -Reverse Engineering Tool and Schema for C++", in ICSM'02, 172-181, 2002

[21] Fluri, B., Würsch, M., Giger, E., and Gall, H., "Analyzing the co-evolution of comments and source code", SQJ, vol. 17/ 4, pp. 367-394., 2009

[22] Gyimóthy, T., Ferenc, R., and Siket, I., "Empirical validation of OOmetrics on open source software for fault prediction", TSE,vol.31/10,Oct' 2005.

[23] Henderson-Sellers, B., Software Metrics, U. K., Prentice Hall, 1996.

[24] Hill, E., Fry, Z. P., Boyd, H., Sridhara, G., Novikova, Y., Pollock, L., and Vijay-Shanker, K., "AMAP: Automatically Mining Abbreviation Expansionsin Programs to Enhance Software Maintenance Tools", in MSR'08.

[25] Kuhn, A., Ducasse, S., and Gîrba, T., "Semantic Clustering: Identifying Topics in Source Code", IST, vol. 49/3, pp. 230-243, 2007

[26] Lawrie, D., Feild, H., and Binkley, D., "Leveraged Quality Assessment Using Information Retrieval Techniques", in ICPC'06, pp. 149-158., 2006

[27] Lawrie, D., Feild, H., and Binkley, D., "Extracting Meaning from Abbreviated Identifiers", in Proc. of 7th IEEE SCAM'07, Paris, France, 2007.

[28] Liu, Y., Poshyvanyk, D., Ferenc, R., Gyimóthy, T., and Chrisochoides, N.,"Modeling Class Cohesion as Mixtures of Latent Topics", in ICSM'09.

[29] Maletic, J. I. and Marcus, A., "Supporting Program Comprehension Using Semantic and Structural Information", in ICSE'01, pp. 103-112., 2001

[30] Marcus, A. and Maletic, J. I., "Recovering Documentation-to-Source-Code Traceability Links using LSI", in ICSE'03, pp. 125-137.

[31] Marcus, A. and Poshyvanyk, D., "The Ideal Cohesion of Classes", inProc. of 21st IEEE ICSM'05, Budapest, Hungary, pp. 133-142., 2005

[32] Marcus, A., Poshyvanyk, D., and Ferenc, R., "Using the Ideal Cohesion of Classes for Fault Prediction in OO Systems", TSE, vol. 34/2, pp.287-300., 2008

[33] Olague, H., Etzkorn, L., Gholston, S., and Quattlebaum, S., "Empirical Validation of Three Software Metrics Suites to Predict Fault-Proneness of Object-Oriented Classes Developed Using Highly Iterative or Agile Software Development Processes", IEEE TSE, vol. 33, no. 6, pp. 402-419., 2007

[34] Poshyvanyk, D., Guéhéneuc, Y. G., Marcus, A., Antoniol, G., and Rajlich,V., "Feature Location using Probabilistic Ranking of Methods based on Execution Scenarios and IR", TSE, vol. 33/6, pp. 420-432., 2007

[35] Poshyvanyk, D. and Marcus, A., "The Ideal Coupling Metrics for Object-Oriented Systems", in IEEE ICSM'06, Phil., PA, pp. 469 - 478., 2006

[36] Poshyvanyk, D., Marcus, A., Ferenc, R., and Gyimóthy, T., "Using Information Retrieval based Coupling Measures for Impact Analysis", Empirical Software Engineering, vol. 14, no. 1, pp. 5-32., 2009

[37] Quah, T.-S. and Thwin, M. M. T., "Application of neural networks for software quality prediction using OO metrics", in ICSM'03,pp. 116-125.

[38] Salton, G. & McGill, M., Introduction to Modern IR, McGraw-Hill, 1983.

[39] Subramanyam, R. and Krishnan, M. S., "Empirical Analysis of CK Metrics for Object-Oriented Design Complexity: Implications for Software Defects", IEEE TSE, vol. 29, no. 4, pp. 297-310., 2003

[40] Tairas, R. and Gray, J., "An Information Retrieval Process to Aid in the Analysis of Code Clones", ESE, vol. 14, no. 1, pp. 33-56., 2009[41] Wilkie, F. G. and Kitchenham, B. A., "Coupling measures and change ripples in C++ application software", JSS, vol. 52, pp. 157-164., 2000