Object Communication Error Prediction in Constructor Development

Abdul Majid Soomro[1], and Bremananth R[2]

**[1, 2] Information Systems and Technology Department,
Sur University College (Affiliated to Bond University, Australia)
P.O. No. 440, PC. Code 411, Sur, Oman.**

Abstract

In dynamic software development, organize the optimization of the usage of resources in order to deliver the product within the finite time is an essential and computational challenging task. Certain robust methods are required to fulfill the requirement of the users to prevent or repair the software faults especially object communications when they are required to inherit in diverse natures. In this paper, an approach for predicting the object creations' run-time errors in the multiple and multi-level of inheritance of objects when larger amount of objects are required to communicate each other. An object fault is concerned with an error due to inheritance and violation of object constraints. A fault prediction model is designed to separate the faulty classes in the field of software testing. Classes are separated according to the inheritance fault encountered in a specific class. Results show that this model can be utilized for predicting software reliability.

Keywords: *Class Inheritance, Fault finding, Object communication, Software Engineering*

1. Introduction

In Object oriented software development, prediction of defects among object communications is an important and challenging software engineering research topic [1]. This paper deals a problem of dynamic software fault prediction of derived classes in the object creation when n number of classes involved in the rapid product development. In the state-of-art-technology, quite number of prediction of software faults are done based on the statistical approaches [2], [3], [4], capture-recapture (CR) models [5], [6], [7], [8], and detection profile methods (DPM) [9] [1]. These methods are utilized to predict the quantity of defects remaining in software systems with assessment of data and course of action of quality data that involved in the software development. The object communication prediction is an important measure for the software developer [10]. It can be used to organize the software process that is to decide whether to schedule further scrutinizes are needed to pass the software artifacts to the next development process or not. Based on the object communications and inheritance property of constructors, a quality of a software system is redelivered [11].

In another work, a set of association rule mining algorithms was suggested from the data mining community to disclose software defect associations [12]. Initially, seeking as many related faults as possible to invoke the faults and consequently make more effective error checking for the software. Based on the literature, software components classified into two major phenomena such as fault-dreary and no-fault-dreary. These two factors can be based on two metrics of classifications, as in state-of-art techniques which are listed out in software engineering references [13]. A fault-proneness was estimated based on the random forests in [14] and T. Menzies et al. utilized data mining strategy based static code attributes to learn the fault predictions in [15]. Certain novel findings in classification models for software defect prediction classification model were suggested by Lessmann [16]. A. Porter et al. suggested a metric-based classification trees for empirical driven software development [17]. K. Ganesan et al. developed a case-based software quality prediction model [18]. A neural network based software quality modeling of larger telecommunications systems was proposed by Khoshgoftaar et al. However, the time complexity for estimation depends on number of objects which are required for the communications in the practice environment [19].

However, ample of works had done for software prediction and classifications. Regrettably, classification of prediction of faulty classes remains a principally unsolved problem. In addition to that, in object oriented paradigm, constructor communication related object faulty functioning is still needed for further research in order to address better solution based on the product development. The erudition of providing solution to these problems may lead to challenges in how the proposed techniques will be configured and how will they validate during the development life-cycle of the software engineering. Shortened or improper validation can produce the result in involuntarily ambiguous. This is one of the reason, we contribute in this paper for a general framework for the object communication and its validity.

From these foundation works, we are motivated to contribute a solution to solve the problem of object communication prediction in constructor development of objected oriented rapid software development. This paper makes the contributions, estimating the run-time object faults during compile-time in order to avoid the catastrophic failure occur in the software running process. Catastrophic errors are encountered when a cascading system failure appears on the software environment. The preliminary work done in this paper is to develop a yet another compiler to use the test classes for the object metrics. The catastrophic faults of classes are predicted based on their constructed behavior when development process is going on the site. They are clustered and output is displayed according to the cluster of the errors in the tested classes. The remainder of this paper is organized as follows: Section 2 described our proposed methodology. Experimental results and analysis for error prediction is depicted in Section 3. Concluding remarks and further enhancements are given in Section 4.

2. Proposed Methodology

Testing object oriented software is a process of significantly increasing reuse, quality, and productivity. There are certain challenging issues in predicting object-oriented software faults such as base class error, external code error, inheritance and dynamic binding. In order to predict the communication fault during object-oriented system development, our proposed framework perform the sequence of steps as shown in Fig. 1. In the first phase, a repository of knowledge base has been formed based two phases such as acquisition of past object's faults, and refinement. Acquisition of source of faults must be of the highest quality in the object communication, else the intellectual of prediction produced downstream. Refinement is the crucial source of value added.solution. It is involved restructuring, relabeling, indexing, and integrating object communication. In addition, refining also refers to cleaning up or sanitizing content so as to ensure complete anonymity of sources and key factor of objects which are involved for the faults. Statistical analyses can be performed on fault content at this stage to conduct a meta-analysis such as pattern found in a collection of communication objects.

Fetch the object from the repository based on the certain pre-fixed rules which are mainly required for extracting the fault knowledge and utilizing them in the present situation of object communication.

2.1 Pre-fixed rules

Object communication fault prediction is essentially involved certain constructor rules which are followed by the traditional object oriented software development. The following are the pre-fixed rule for the prediction process.

- Constructor's object can use any access modifiers.
- If the private constructor's object wants to allow an instance of the class to be used, then the class must provide a static method or variable that allows access to an instance created from within the class [11].
- The constructor object name must match the name of the class.
- Constructor must not have a return type.
- Constructor with no argument is the default constructor.
- If programmer doesn't include constructor into the class, then a default constructor will be automatically generated by the compiler.
- Abstract classes may have constructors, and those constructors are always referred when a concrete subclass is instantiated.
- If software engineer uses private access modifier for constructor, then we need to instantiate the object within the class code itself.
- A call to super class can be either a no argument call or can include arguments passed to the super constructor.
- Software engineer cannot make a call to an instance method, or access an instance variable, until after the super class constructor runs.

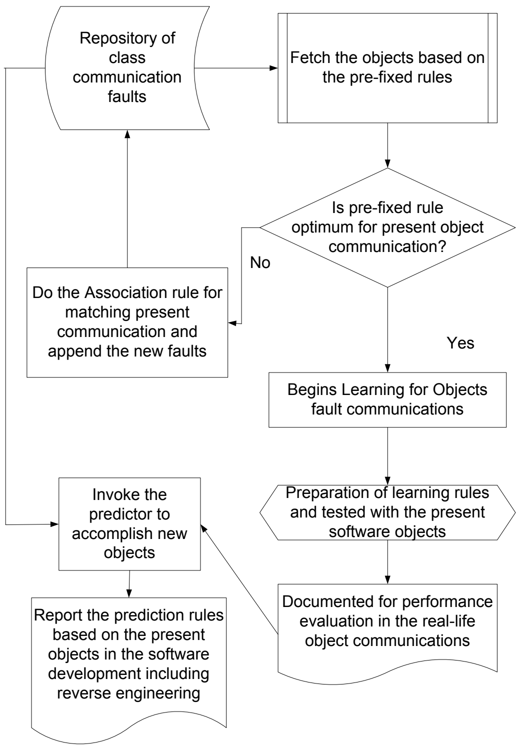
Repository of class communication faults
Fetch the objects based on the pre-fixed rules
Is pre-fixed rule optimum for present object communication?
No
Yes
Do the Association rule for matching present communication and append the new faults
Begins Learning for Objects fault communications
Preparation of learning rules and tested with the present software objects
Invoke the predictor to accomplish new objects
Documented for performance evaluation in the real-life object communications
Report the prediction rules based on the present objects in the software development including reverse engineering

Fig.1 Representation of sequence of operation involved in the object communication prediction process.

2.2 Optimum constraint

Based on the object communication pre-fixed rule base, the ongoing scenario of software development is checked for optimum criterion. Let N objects are being involved for communication in the development process, then their optimum constraints are evaluated based on the optimum soft-decision. Since, there is no specific criterion for constructor derivation the log likelihood ratio (LLR) will be utilized for the optimum condition checking. Based on the present scenario, making the decision for present object communication is crucial factor. LLR is described as

$$L_o = \log\left(\frac{\sum_{1...N} e^{1/\sigma^2 (x-f_x)^2}}{\sum_{1...N} e^{1/\sigma^2 (y-f_y)^2}}\right) \quad (1)$$

where L_o denotes an optimum log likelihood ratio (LLR), N represents number of objects which are involved for the communication. The parameters σ^2, x, y, f_x and f_y denote variance of fault communication, index of fault knowledge base units in x-direction, index of fault knowledge base units for non-fault classes in y-direction, fault communication and non-fault communication.

2.3 Association rule computation

Software errors, formal specification and design changes are primarily causes of disproportionate cost and rescue of a software project. Certain works were proposed for association rule mining [20]. Amasaki et al. [21] suggested a set of association rule mining with preconditions for software risk assessments.

Let $E = \{e_1, e_2, ..., e_m\}$ be a set of errors which are encoded in the form of binary relationship as shown in Table I. In real-time software development, there is possibility that an initiates subsequent of errors. Especially object communications, if any object in a constructor causes error, then subsequent communication will lead the sequences of other sources of errors. For example, we can represent that $\beta \wedge \lambda \Rightarrow \eta$ is implication of errors β and λ initiate another error is called η. In general, a set $\eta \subset E$ is called the subset of existing repository of errors. Let an error repository ϖ consists of multi-set of errors in E. Each invoking of error communication $\kappa \in \varpi$ called an error-maneuver. The format of the association rule is implied that $\beta \Rightarrow \eta$ and $\beta \cap \eta = \phi$.

Table 1: Representation of errors, encoding and priority level of association relationship.

Error Encoding	Description of Errors	Priority
0001	Constructors object is not properly used for access modifiers.	0
0010	The class might not provide a static method or variable.	1
0011	The constructor object name might not match the name of the class.	2
0100	No argument constructor.	3
0101	Default constructor will be automatically anonymously.	4
0110	Abstract classes may not have constructors	5
0111	No instantiate the object within the class code itself.	6
1000	A call to super class can be no argument call.	7
1001	Cannot make a call to an instance method.	8

Our method of computing association rules is a sequence of binary relationships between the classes. For example, relation between derived classes and test case classes will be represented as $\{d, t_c\}$. If consider the errors between d and t_c can be computed, then association of binary relationship will be obtained. Perhaps, in a software development, if n classes will be involved, then $n*(n-1)$ error association rules are obviously required to resolve the communication errors between them. In addition, the inverse of error association rules will be also included. However, the probability of errors is involved in the software development that can be represented using diverse metrics. First metrics is called support that can be represented as $s(d \Rightarrow t_c)$. The probability of $s(d \Rightarrow t_c)$ will be $p(d,t_c)n$. It denotes the counts of implications of error in communications. For example, assume that the total number of errors in the software class communications will be n = 32 for a software development. The number of errors that includes $s(d, t_c)$ will be 8, and then $s(d \Rightarrow t_c)$ will be 0.25. The ratio of the number of errors in communications that contain $d \cup t_c$ to the number of errors in d is referred as confidence. This is described as

$$c(d \Rightarrow t_c) = p(d \mid t_c) = s(d, t_c)/s(d) \ . \quad (2)$$

Where $c(d \Rightarrow t_c)$ and $p(d \mid t_c)$ represent confidence metric and probability of confidence metric, respectively. Support of d is dented as $s(d)$. However this measure is not symmetrical.

$I(d \Rightarrow t_c)$ is a metric to compute the correlation between two error objects called d and t_c. This is called interest. It represents how many epochs more often object d and t_c are contained in a commit communication then anticipated if they are statistically self-determining. However, this metric is symmetrical.

$c_o(d \Rightarrow t_c)$ denotes a metrics of conviction. It represents a metrics of the implication that whenever object d is committed error and object t_c is also committed errors. This described as

$$c_o(d \Rightarrow t_c) = p(d)p(\neg t_c)/p(d,\neg t_c) \; . \qquad (3)$$

The implementation of $I(d \Rightarrow t_c)$ and $c_o(d \Rightarrow t_c)$ are described as

$$I(d \Rightarrow t_c) = s(d,t_c)n/s(d)s(t_c) \, . \qquad (4)$$

$$c_o(d \Rightarrow t_c) = (s(d)n - (s(d)s(t_c))/n)/s(d) - s(d,t_c) \; . \qquad (5)$$

Based on these metrics, if the pre-fixed rule is optimum then learning for object fault communication is begun in order to prepare the learning rules to test with the present software objects.

2.4 Learning for Object fault Communication

In the object fault learning process, the states of each object communication is verified. This is a pre-process of preparation of learning rules for software testing. During testing practices, we categorize association learning rules according to the errors, constructor code, test code and invocation of server and client methods. Fig. 2 shows a transition diagram for the error learning normally occurred in the class creation.

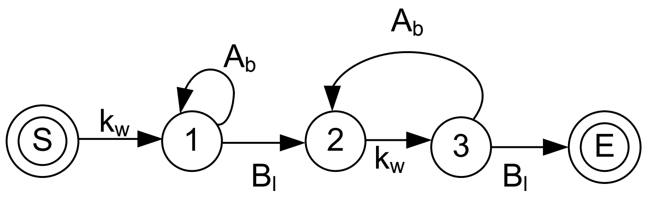

Fig. 2 Representation of state transition for the learning of fault communication.

In the aforesaid figure, transition State S starts from double circle by invoking an object creation keyword k_w and goes to State 1 followed by alphabets for object names. This is represented by the self-loop in State 1. A set of blank white space is required in order to act as a delimiter for an object name as denoted as State 2. Once super class keyword (for example, *extends* in Java) is invoked perfectly then its corresponding name existences will be learned. This name space may be a part of software system package (for example, Applet) or any user defined name based on the software projects. However, learning of every objects and classes name is necessary for entire software testing. Furthermore, a double state self loop illustrated in Fig. 1 represents that certain object communication has a multithreading or interface keywords. However, it is not always the case for the software development. Once multithreading or any number of interfaces are invoked then sequences of name of its classes or interfaces given by the software engineer. Systematic learning is an important process based on which programming language will be utilized for the software development. Fig. 3 shows a state transition diagram for creation of constructor in the object communication process. It has the labels of A_c, C_m, and C_c denote derived & super class name, main function object and constructor creation.

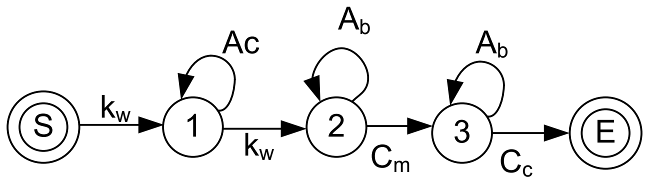

Fig. 3 State transition for the constructor creation.

State S is started with keyword k_w and it goes to State 1. In this State, an object of the class is created. Its self-loop depicts object name symbolized by A_c. By another keyword, it goes to State 2. It has a self-loop of alphabets. From this state, main function object C_m is invoked by State 3 and it has self-looped for the alphabets and ends with constructor creation C_c. Learning for object faults of each state has been performed based on the k_w is invoked. There is no k_w is invoked, then the system state is called to make an error transition. State 2 repeats by itself with alphabets until name is not completed then make transition from State 2 to 3 when call main invoked. After that it makes transition from State 3 to 4 with new constructor and State 4 ends.

2.5 Object Communication Error Learner Construction

A learning scheme consists of a class error preprocessing, an attribute selection, and a learning algorithm. Class error preprocessing is a significant part of constructing a practical learner. In this phase, the learning object fault communication are preprocessed, such as removing unwanted blanks, handling missing alphabets, and decoding errors based on the priority level. Learning algorithm requires the best parameters for learning the diverse error data. Even though all the parameters are helpful for performing learning, a few of the parameters may spin-off the error prediction process. Hence, selection of optimal parameter for the learning algorithm plays vital role for the error prediction process which performs learning process on repository of data. In our proposed method, binding parameter selection is employed for parameter selection. This is a computationally expensive however it gives better result for prediction of error. Binding parameter selection performs annoyed-rationale in order to assess the prominence of different parameter subsets. Once parameter selection is completed, then preprocessed error data are abridged to the best parameter detachment. Thereafter, abridged error exercise data and learning algorithm are utilized for constructing object communication error learner (OCEL). OCEL is tested prior to the new set of object errors preprocessed and its facet has been abridged to the best detachment of parameters. The estimated value of OCEL is put side by side with substantial value of errors of the tested data, and then the recital of predictor is evaluated. Its validation is obtained based on the statistical behavior of the OCEL. Based on the prominence of the learner, error predictor is invoked in order accomplish the activity of new object communication.

2.6 Reporting prediction rule and Reverse Engineering

Once OCEL is chosen according to its testimony, then its related prediction rules are named to the software engineer for assisting their code amendment. Predictor is invoked by the stricture such as precedent error objects in the repository, new error object communication and OCEL. As stated previously, learning will provide a present appropriate predictor and its optimal parameters. An innovative set of parameters are chosen based on the new set of error objects. These new optimal parameters are utilized for the present appropriate predictor for estimating the result of communication errors among the objects. Furthermore, as a part of reverse engineering, object are newly incorporated at the time of system release due the factors of serious system fault, plate form change, or competing with other software products. These object communication errors will be predicted during reverse engineering. This is a challenging process and certain works has been done previously [23]. Reverse Engineering is an entrenched practice in that there are fewer Computer aided software engineering (CASE) tools available to identify software release errors and convert reverse into a good quality structural models. A frame work for predicting a possible error during reverse engineering is described as shown in Fig. 4.

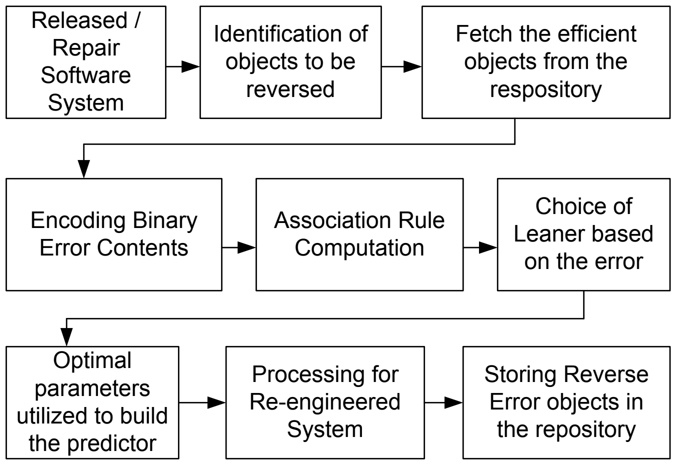
Released / Repair Software System
Identification of objects to be reversed
Fetch the efficient objects from the respository
Encoding Binary Error Contents
Association Rule Computation
Choice of Leaner based on the error
Optimal parameters utilized to build the predictor
Processing for Re-engineered System
Storing Reverse Error objects in the repository

Fig. 4. A framework for Object communication error prediction in reverse software engineering process.

In the reverse process, final software will be revoked into a design process. First, reversed objects are identified and the respective efficient objects are fetched from the repository based on the minimum matching rule. Once object is fetched, then error encoding process will be carried out and its association rules are computed. Prediction of errors in object communication is based on the efficiently choosing the leaner. The leaner is trained and tested with the reversed objects before incorporated into structure model of the software engineering. Optimal parameters are selected based on the how they are useful for the error processing. Predictor is designed for the reverse engineering error prediction based on the chosen optimal parameters. Tested objects are validated with predictor in order to estimate the errors before fixing into the model. Upon successful of incorporating the objects the experienced error scenario of objects' communication will be stored in the repository.

3. Experimental Results and Analysis

The performance of the proposed framework for object communication error prediction is validated using 150 different kinds of object communication errors usually occur in the software product development. The predictor is trained and tested with the error data set. A common metric for predictor quality evaluation called Receiver Operating Characteristics (ROC) is utilized for checking the object data sets. For each categories of error, a predictor is trained and its thresholds across the hiatus between 0 and 1 are pertained to the results of the predictor. For each threshold, two set of metrics are computed such as True Positive Ratio (TPR, ψ_t) and False positive Ratio (FPR). Object communication errors prediction for TPR is computed using (6)

$$\psi_t = \frac{\kappa}{\kappa + \vartheta}, \tag{6}$$

where ψ_t, κ and ϑ denote true positive ratio, true positive (TP) and false negative (FN), respectively.

Error prediction for False positive Ration (FPR, ρ_t) is described as.

$$\rho_t = \frac{\upsilon}{\upsilon + \nu}, \tag{7}$$

where ρ_t , υ and ν represent false positive ratio, false positive (FP) and true negative (TN), respectively.

Figure 5 shows the ROC for the proposed framework to predict diverse errors possibly occurred in the object communication errors. The training ROC is mainly for evaluating the training behavior of the predictor as shown in Fig. 5a. This shows EER that represents equal error rate (EER). It is a performance measure that denotes a superlative position where predictors recognize all the object communication error without any further bugs. Furthermore a higher probability of error prediction is represented by green color legend and red color legend depicts lower probability of no error prediction by the predictor. However the values in the ROC between 0 and 1 are a superlative position where all the predictors are performed well. In addition, EER should be determined the best among a set of predictors and evaluating its best validation performance its mean squared error (MSE) is also included. As shown is Fig. 6, best validation performance of a predictor is 0.15909 at the epoch of training the leaner at 23. The validation of ROC is shown is Fig. 5b. OCEL testing process of predictor is observed and plotted as shown in Fig. 5c. In our experiment, we study that diverse result for the same predictor can obtain when different discrimination thresholds are utilized. In the representation of ROC, for each threshold, when TPR is computed, the number of prediction of the predictor is greater or equal to the threshold and divided by the number of successful prediction of errors. All predictors' errors prediction and its ROC are represented in Fig. 5d.

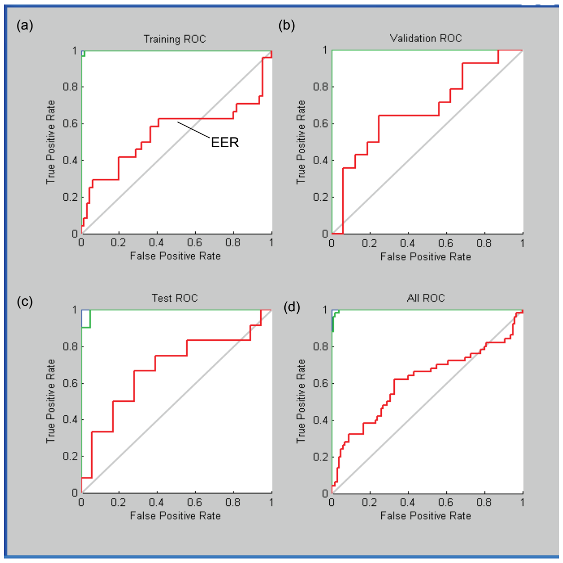

Fig. 5 Representation of ROC (a) Training ROC (b) Validation ROC (c) Testing ROC (d) Combination of all ROC.

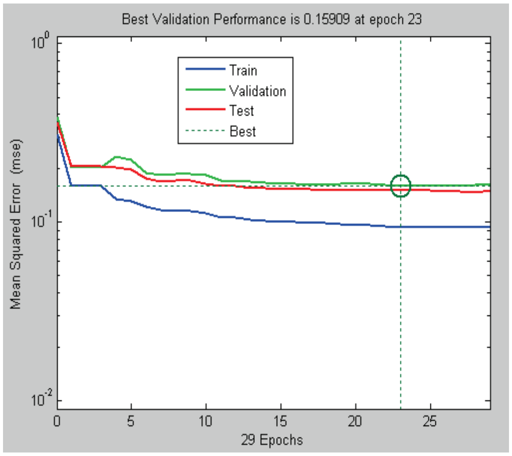

Fig. 6 Mean Squared Error Computation for the Predictor.

In Fig. 5d, each value of ROC is mapped to the respective bias thresholds. ROC at $p(\psi_t)=0$ and $p(\rho_t)=0$ denote that the predictor delights all error object communication as non-error and explicitly the corresponding threshold is one. At value between $p(\psi_t)=1$ and , $p(\rho_t)=1$, the predictor delights all as error and explicitly the corresponding threshold is zero. Thus, the ROC exemplifies the performance of the predictor among the threshold variations. The gradient and validation checks are studied for the predictor and value of gradient was 0.0046306 at 29th epoch. This is shown in Fig. 7.

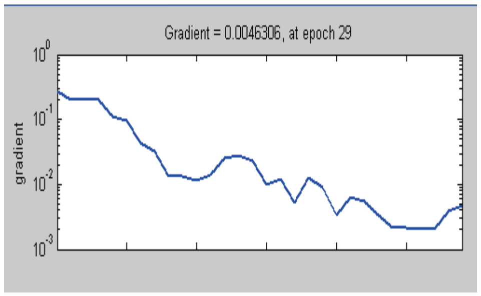

Fig. 7 Illustration of gradient.

4. Conclusions

This research paper proposes a frame work for the object communication error prediction among constructor implementation. It involves a sequence of steps such as fixing rule for the prediction of constructor errors, optimum constraint selection based on the likelihood ratio, association rule computation and selection of leaner based on the prediction. In our approach diverse leaning methods are examined for the new object faults and the most optimal leaner has been chosen based on the errors in the communication. In addition parameters of the learner is estimated in order to construct a predictor and tested with both existing repository and new set of data. Another contribution of reverse engineering based object error communication problems has been addressed and a framework is proposed. The proposed reverse engineered framework will be easily incorporated into the existing software model. As these frameworks progressively afford more intelligent prediction of errors especially in the object communications, the proposed work can be predicted objects efficiently and portend for object oriented society.

We have done experiments to scrutinize the behavior of the prediction process of software models. The performance evaluation is performed based on the ROC and MSE. Association rule computation was evaluated using metrics such as support, confidence and correlation. The results of training, validation, tested new set of data represented that the proposed scheme of prediction and parameter selection provide good nature of fault prediction for the object oriented paradigms of software development. Furthermore, reverse engineering object faults communications are also predicted using the proposed framework. In near future, a human computer interaction based interactive model will be suggested to enhance the prediction process of diverse area of fault prediction especially critical system software development.

Acknowledgments

Authors would like to thank the Chairman, Dean and The Management, Sur University College to provide financial assistant to do this research a successful manner.

References

[1] Qinbao Song, Zihan Jia, Martin Shepperd, Shi Ying, and Jin Liu, "A General Software Defect-Proneness Prediction Framework," IEEE Trans. On Soft. Eng. vol 37, No. 3, May/June 2011.

[2] B. T. Compton and C. Withrow, "Prediction and Control of ADA Software Defects," J. Systems and Software, vol. 12, no. 3, pp. 199-207, 1990.

[3] J. Munson and T. M. Khoshgoftaar, "Regression Modelling of Software Quality: Empirical Investigation," J. Electronic Materials, vol. 19, no. 6, pp. 106-114, 1990.

[4] N. B. Ebrahimi, "On the Statistical Analysis of the Number of Errors Remaining in a Software Design Document After Inspection," IEEE Trans. Software Eng., vol. 23, no. 8, pp. 529-532, Aug. 1997.

[5] S. Vander Wiel and L. Votta, "Assessing Software Designs Using Capture-Recapture Methods," IEEE Trans. Software Eng., vol. 19, no. 11, pp. 1045-1054, Nov. 1993.

[6] P. Runeson and C. Wohlin, "An Experimental Evaluation of an Experience-Based Capture-Recapture Method in Software Code Inspections," Empirical Software Eng., vol. 3, no. 4, pp. 381-406, 1998.

[7] L. C. Briand, K. El Emam, B.G. Freimut, and O. Laitenberger, "A Comprehensive Evaluation of Capture-Recapture Models for Estimating Software Defect Content," IEEE Trans. Software Eng., vol. 26, no. 6, pp. 518-540, June 2000.

[8] K. El Emam and O. Laitenberger, "Evaluating Capture-Recapture Models with Two Inspectors," IEEE Trans. Software Eng., vol. 27, no. 9, pp. 851-864, Sept. 2001.

[9] C. Wohlin and P. Runeson, "Defect Content Estimations from Review Data," Proc. 20th Int'l Conf. Software Eng., pp. 400-409, 1998.

[10] G. Q. Kenney, "Estimating Defects in Commercial Software during Operational Use," IEEE Trans. Reliability, vol. 42, no. 1, pp. 107-115, Mar. 1993.

[11] F. Padberg, T. Ragg, and R. Schoknecht, "Using Machine Learning for Estimating the Defect Content After an Inspection," IEEE Trans. Software Eng., vol. 30, no. 1, pp. 17-28, Jan. 2004.

[12] N. E. Fenton and M. Neil, "A Critique of Software Defect Prediction Models," IEEE Trans. Software Eng., vol. 25, no. 5, pp. 675-689, Sept./Oct. 1999.

[13] Q. Song, M. Shepperd, M. Cartwright, and C. Mair, "Software Defect Association Mining and Defect Correction Effort Prediction," IEEE Trans. Software Eng., vol. 32, no. 2, pp. 69-82, Feb. 2006.

[14] L. Guo, Y. Ma, B. Cukic, and H. Singh, "Robust Prediction of Fault-Proneness by Random Forests," Proc. 15th Int'l Symp. Software Reliability Eng., pp. 417-428, 2004.

[15] T. Menzies, J. Greenwald, and A. Frank, "Data Mining Static Code Attributes to Learn Defect Predictors," IEEE Trans. Software Eng., vol. 33, no. 1, pp. 2-13, Jan. 2007.

[16] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch, "Benchmarking Classification Models for Software Defect Prediction: A Proposed Framework and Novel Findings," IEEE Trans. Software Eng., vol. 34, no. 4, pp. 485-496, July/Aug. 2008.

[17] A. Porter and R. Selby, "Empirically Guided Software Development Using Metric-Based Classification Trees," IEEE Software, vol. 7, no. 2, pp. 46-54, Mar. 1990.

[18] K. Ganesan, T. M. Khoshgoftaar, and E. Allen, "Case-Based Software Quality Prediction," Int'l J. Software Eng. and Knowledge Eng., vol. 10, no. 2, pp. 139-152, 2000.

[19] T. M. Khoshgoftaar, E.B. Allen, J. P. Hudepohl, and S.J. Aud, "Application of Neural Networks to Software Quality Modeling of a Very Large Telecommunications System," IEEE Trans. Neural Networks, vol. 8, no. 4, pp. 902-909, July 1997.

[20] Agrawal R., Imielinski T., Swami A.,"Mining Association Rules between Sets of Items in Large Databases," In Proceedings of ACM SIGMOD Conference on Management of Data, pp. 207-216, 1993.

[21] Amasaki S., Hamano Y. , Mizuno O. , and Kikuno T. ,"Characterization of Runaway Software Projects Using Association Rule Mining," In Proceedings of 7th International Conference on Product Focused Software Process Improvement, pp.402-407, 2006.

[22] Bremananth R, Thushara R, "Fault Predictions in Object Oriented Software," International Journal of Computer Science and Engineering, vol.1 no.2, pp. 81-88, 2009.

[23] Jahnke J.H, Walenstein A., "Reverse Engineering tools as Media for Imperfect Knowledge," Proc. Of IEEE, Working Conference in Reverse Engineering, 2000, pp. 22 – 32, 2000.

Abdul Majid Soomro received the M.Sc. degrees in Computer Science from Bahu Din Zakriya University Multan Pakistan in 1998 he has completed his Becholer degree in Science from Bahu Din Zakrya University Multan in 1995, Pakistan.

He worked as Senior Lecturer in Preston University, Pak Lawrence institute, Scholar Group of colleges in Pakistan. He also worked as Senior Lecturer with Pakistan School/College Salalah Oman. Currently, he is working as Faculty member of Information Systems and Technology Department at Sur University College (SUC), Oman.

Bremananth R received the B.Sc and M.Sc. degrees in Computer Science from Madurai Kamaraj and Bharathidsan University in 1991 and 1993, respectively. He obtained M.Phil. degree in Computer Science and Engineering from Government College of Technology, Bharathiar University, in 2002. He received his Ph.D. degree in 2008 from Department of Computer Science and Engineering, PSG College of Technology, Anna University, Chennai, India. He has completed his Post-doctoral Research Fellowship (PDF) from the School ofElectrical and Electronic Engineering, Information Engineering (Div.) at Nanyang Technological University (NTU), Singapore, 2011. Before joining NTU, Singapore, he was a Professor and Head, Department of Computer Science and Application, in India. He has 18+ years of experience in teaching, research and software development at various Institutions. Currently, He is an Assistant Professor for Information Technology at Information Systems and Technology Department, Sur University College, Sur, Oman, affiliated to Bond University Australia. He is an associate editor of various International Journals in USA and He is an active reviewer of various IEEE International conferences/Journals. His fields of research are Acoustic holography, Acoustic imaging, Pattern recognition, Computer vision, Image processing, Biometrics, Multimedia, Computer network, Software engineering, Soft computing and Microprocessors.

Dr. Bremananth received the M N Saha Memorial award for the BestApplication Oriented paper in the year 2006 by Institute of Electronics and Telecommunication Engineers (IETE). His continuous contribution of research was recognised by Who's who in the world, USA and his biography was published in the year 2006. He is a member of Indian society of Technical Education (ISTE), Advanced Computing Society (ACS), International Association of Computer Science and Information Technology (IACIT) and Institute of Electrical and Telecommunication Engineers (IETE).