International Journal of Enhanced Research in Management & Computer Applications, ISSN: 2319-7471
Vol. 3 Issue 2, February-2014, pp: (6-11), Impact Factor: 1.147, Available online at: www.erpublications.com

An efficient Software Quality Estimation: Critical Review

Mr. Asif Ali[1], Dr. Kavita Choudhary[2], Dr. Ashwini Sharma[3]
[1]Ph.D Scholar, [2,3]Assistant Professor
[1,2]Dept. of Computer Science & Engineering, Jagannath University (Raj.), India
[3]Dept. of Computer Science & Engineering, GEC Jhalawar (Raj.), India

Abstract: In the software Quality to predict the good quality of the programs or the software module is required for software quality estimation. This paper represents that track and detection of potential software faults near the beginning, which is critical in many high assurance systems. On the other hand, creating such accurate quality-estimation design is challenging because the data with noise and unwanted data usually degrades the performance of the whole system or the model. After studying several research woks in this area, we come across with two main problems. First is the noise and second which is the main problem is the parameter on which you can categorize the programs for the betterment. So our survey papers concerted in the two relevant directions and find the pros and cons in the existing traditional techniques. Development practitioners typically construct quality classification or fault prediction models using software metrics and fault data from a previous system release or a similar software project. To establish a proper Software Measurement Methods need a clear understanding of the appropriate data collection, analysis and reporting requirements also. It can be categorized by different accuracy measurement of software model like F-Measure, Odd Ratio and Power. Based on the discussion we also suggest some future enhancements which are useful in software quality estimation trends.

Keywords: Encryption, DES, Security Measures, RSA.

1. Introduction

Software similar to carve rear is worn to brand program modules that are likely to be defective [1], [2][3]. Afterwards, the restrictive dogmatic allocated for software make public corroboration and beyond part be targeted approaching those program modules, achieving a cost-effective resource utilization [4]. A software presence merit grave allows the software abet consummation to run after and learn of capability software defects take a part in early on during development, which is critical to many high-assurance systems[5][6].

To predict about the good quality of the programs or the software module is necessary. This lets them track and detect potential software faults early on, which is critical in many high assurance systems. However, creating such accurate quality-estimation design is challenging because the data with noise data usually degrades the performance of the whole system or the model. After studying several research woks in this area, we come across with two main problems. First is the noise and second which is the main problem is the parameter on which you can categorize the programs for the betterment [7][8].

There are generally two types of noise in the category of data quality and software parameters. The first is concern with the mislabeled software modules, caused by software engineers failing to detect, forgetting to report, or simply ignoring the existing software faults. Removing such noisy instances can significantly improve the performance of calibrated software quality-estimation models [9][10]. Another main challenge is that, in real-world software projects, we need to find the parameters on that we can estimate the quality. So there is the prediction strategy on the basis we can learn and predict. The remaining of this paper is organized as follows. In Section 2 we discuss about software quality estimation techniques. In section 3 we discuss about the literature review. Problem formulation **mentioned** in section 4. Analysis is given in section 5. The conclusions and future directions are given in Section 6. Finally references are given.

2. Software Quality Estimation Techniques

To establish a proper Software Measurement Methods need a clear understanding of the appropriate data collection, analysis and reporting requirements. The starting point is the identification of the audiences for measurement and their unique needs. We also identify the key measures, source of data, analyze and interpret the metrics, report the information parameters, available testing tools and how we apply the ongoing process on the available tools. So our measurement framework accomplishes mainly four things 1) Metrics 2) Analysis 3) Final Qualification 4)

Improvement Analysis. There are several researchers which has devoted much research to developing and studying effective software metrics that characterize a software system's complexity. Those researchers and practitioners have analyzed and provide some study with several software complexity metrics, most of them are used for program code for expressing software complexity. These software metrics have been used widely worldwide and successfully used even though some issues related to their effectiveness remain open research problems.

In 2000, Briand et al. [11] provide a logistic regression which is based on object oriented parameters to explore the relationships between four groups of OO metrics and the fault-detected system classes during testing. Means the investigation is based on object oriented programming, so we can extend with other testing approaches.

In 2003, Reformat et al. [12] exploited several techniques of computational intelligence to support the quality assessment of individual software objects. These techniques covered granular computing, neural networks, self-organizing maps and evolutionary-based developed decision.

In 2007, Kanmani et al. [13] investigated two fault prediction models based on OO metric and neural networks using a dataset of software modules designed by students. Among the two neural networks, the probabilistic neural network outperformed the back propagation one in accurately predicting the fault-proneness of the OO modules. Based on the above investigation we can select Object Oriented Modules for testing. There are several work is done in this direction by using neural network and Bayesian theorem. Software clustering is needed because we want to reduce the overhead of data processing.

A generic re-engineering source code transformation framework to support the incremental migration of procedural legacy systems to object-oriented platforms is. First, a source code representation framework that uses a generic domain model for procedural languages allows for the representation of Abstract Syntax Trees as XML documents. Second, a set of transformations allow for the identification of object models in specific parts of the legacy source code. In this way, the migration process is incrementally applied on different parts of the system. A partitioning algorithm is used to decompose a program into a set of smaller components that are suitable for the incremental migration process. Finally, the migration process gradually composes the object models obtained at every stage to generate an object model for the whole system.

The F measure is then a measure of the algorithms precision and recall.

F= (2 * precision * recall) / (precision + recall), where: Precision (P) = cells correctly put into a cluster / total cells put into the cluster Recall (R) = cells correctly put into a cluster / All the cells that should have been in the cluster.

The odds ratio is a measure of effect size, describing the strength of association or non-independence between two binary data values. It is used as a descriptive statistic, and plays an important role in logistic regression.

OR= 2* Recall (1-Precision)/(1-Recall*Precision)
Power (PO) is defined as:
PO= ((1-Precision) k-(1-Recall)k)

3. Literature Survey

In 2009, Mark Shtern[15] discuss about several software clustering algorithms Most of these algorithms have been applied to particular software systems with considerable success. However, the question of how to select a software clustering algorithm that is best suited for a specific software system remains unanswered. They introduce a method for the selection of a software clustering algorithm for specific needs. The proposed method is based on a newly introduced formal description template for software clustering algorithms. Using the same template, we also introduce a method for software clustering algorithm improvement.

In 2010, Ramandeep S. Sidhu [16] uses subtractive clustering based fuzzy inference system approach which is used for early detection of faults in the function oriented software systems. This approach has been tested with real time defect datasets of NASA software projects named as PC1 and CM1. Both the code based model and joined model of the datasets are used for training and testing of the proposed approach. The performance of the models is recorded in terms of Accuracy, MAE and RMSE values. The performance of the proposed approach is better in case of Joined Model. As evidenced from the results obtained it can be concluded that Clustering and fuzzy logic together provide a simple yet powerful means to model the earlier detection of faults in the function oriented software systems.

In 2010, Mark Shtern [17] introduces and quantifies the notion of clustering algorithm comparability. It is based on the concept that algorithms with different objectives should not be directly compared. Not surprisingly, we find that several of the published algorithms in the literature are not comparable to each other.

In 2010, Jin-Cherng Lin et al. [18] suggest majority of development teams will feel time isn't enough to use or the project valuation be false to make the software project failed. However the cost of the software project is almost a manpower cost, manpower cost and then become a direct proportion with development schedule, so precise effort the valuation more seem to be getting more important. One-way analyze to select several factors then used K-Means clustering algorithm to software project clustering. After project clustering, they use Particle Swarm Optimization that take mean of MRE (MMRE) as a fitness value and N-1 test method to optimization of COCOMO parameters. Finally, take parameters that finsh the optimization to calculate the software project effort that is want to estimation. This research use 63 history software projects data of COCOMO to test. The experiment really expresses using base on project clustering with multiple factors can make more effective base on effort of the estimate software of COCOMO's three project mode.

In 2011, Rashid Naseem et al. [19] analyze the Russell and Rao measure for binary features to show the conditions under which its performance is expected to be better than that of Jaccard. They also show how our proposed Jaccard-NM measure is suitable for software clustering and propose its counterpart for non-binary features. Experimental results indicate that their proposed Jaccard-NM measure and Russell & Rao measure perform better than Jaccard measure for binary features, while for non-binary features, the proposed Unbiased Ellenberg-NM measure produces results which are closer to the decomposition prepared by experts.

In 2011, Ural Erdemir et al. [20] understanding a software system is not an easy task because in most cases documentation of software design is outdated, incomplete or absent. Therefore support of tools and algorithms are necessary for software developers to understand software quicker and easier. Clustering algorithms have been widely used for software architecture recovery. Their performance depends not only on the algorithm itself but also on the nature of the software system. They propose the adaption of the fast community detection algorithm for object-oriented software clustering and evaluate its performance with other clustering algorithms in the literature. It is an agglomerative hierarchical clustering algorithm that has been introduced to find communities in networks. The algorithm can operate on directed weighted graphs and it has a considerable speed advantage over other algorithms.

In 2012, Árpád Beszédes et al. [21] report on a complex project involving industrial partners whose aim is the development of a unified software quality platform that deals with and bridges these low and high level quality aspects, and provides a basis for the industrial applications of the approach. The project is implemented by a consortium of software industry members of the Szeged Software Innovation Pole Cluster and associated researchers with the support from the EU co-financed national grant promoting innovation clusters of small and middle-sized enterprises. The approach to the unified quality platform is based on the Goal-Question-Metric paradigm and a supporting software infrastructure, and its novelty lies in a unified representation of the low level metrics and the high level questions that evaluate them to address software quality assurance goals. Information which is related to the design and development of the quality platform and it is also related to the applications that are being developed by the industrial members of the consortium.

In 2012, Deepak Gupta et al. [22] discusses about clustering which is the unsupervised classification of patterns into groups. A clustering algorithm partitions a data set into several groups such that similarity within a group is larger than among groups the clustering problem has been addressed in many contexts and by researchers in many disciplines; this reflects its broad appeal and usefulness as one of the steps in exploratory data analysis. There is need to develop some methods to build the software fault prediction model based on unsupervised learning which can help to predict the fault –proneness of a program modules when fault labels for modules are not present.

In 2012, Puneet Dhiman et al. [23] Software defects plays important role to take the decision about when the testing will be stopped. Software defects are one of the major factors that can decide the time of software delivery. Not only has the number of defects also the type of defect as well the criticality of a software defect affected the software quality. Software cannot be presented with software defects. All the Software Quality estimation approaches like CMM etc. follow the software defects as a parameter to estimate the software quality. We are trying to categorize the software defects using some clustering approach and then the software defects will be measured in each clustered separately. Their proposed system will analyze the software defect respective the software criticality and its integration with software module.

In 2013, A. Charan Kumari et al. [24] presents a Fast Multi-objective Hyperheuristic Genetic Algorithm (MHypGA) for the solution of Multiobjective Software Module Clustering problem. Multi-objective Software Module Clustering

Problem is an important and challenging problem in Software Engineering whose main goal is to obtain a good modular structure of the Software System. Software Engineers greatly emphasize on good modular structure as it is easier to comprehend, develop and maintain such software systems. In recent times, the problem has been converted into a Search-based Software Engineering Problem with multiple objectives. This problem is NP hard as it is an instance of Graph Partitioning and hence cannot be solved using traditional optimization techniques. The MHypGA is a fast and effective metaheuristic search technique for suggesting software module clusters in a software system while maximizing cohesion and minimizing the coupling of the software modules. It incorporates twelve low-level heuristics which are based on different methods of selection, crossover and mutation operations of Genetic Algorithms. The selection mechanism to select a low level heuristic is based on reinforcement learning with adaptive weights.

4. Problem Domain

After discussing several research works we can come with some problem area in the traditional approaches which are following:

1) There are several clustering algorithm but which algorithm is better for the condition specified is a major issue. How can we determine it, is also a big challenge.
2) Any approach in the direction of better software module identification should be applicable to different domains.
3) Classification is also missing in the related attribute relationship which can be a better in fault prediction.
4) Need of a Hybrid framework where we employ decomposition, incremental delivery, identification and organization.
5) Presenting the software data at a finer granularity and analyzing other software
systems using this granularity with some
Semi-supervised classification schemes to facilitate minimal amount of expert involvement will be better reengineering.
6) Dynamic dependencies, documentation, bug reports, software metrics and preprocessing techniques are essential for software clustering [25].
7) Modularization or software reuse will be helpful.
8) We can involve more modularity views and provide more comprehensive deviation trend monitoring for evolution decisions [26].
9) Partition algorithm will be used for reducing the overhead generated by the large database.
10) Combining dynamic analysis with system's structural model, e.g. program-element dependencies for enriching the semantic information and erecting the metric evaluation for different association rules, and setting right weights for each edge correlation, and conducting systematic empirical studies on the choice of the appropriate parameters for the proposed approach [27].
11) Part of the parameter optimization algorithm can further explore whether there is more suitable algorithm for software engineering issue of estimate software development effort or using new algorithm to optimize the parameter [18].

5. Analysis

After studying and observing several research works we compare the result discussions by their techniques, so that we identify the good and flaws presented in the previous research.

Table 1: Analysis

S.no	Approach	Average Data Rates	Results
1	Mining Association Rules to Facilitate Structural Recovery[27]	70 %	Proposing an approach for program comprehension through association rule mining analysis and demonstrating the visualization technique comparing the dependences within clusters and system entities [27].
2	Software Project Clustering[18]	0.6187 Four Group	Multiple factors can make more effective base on effort of the estimate software of COCOMO's three project mode.

3	Software Project Clustering[18]	0.5459 Three Group	
4	Similarity Measures[19]	When feature vector has d = 0, then Jaccard-NM and Russell & Rao become equal to Jaccard measure. Jaccard-NM and Russell & Rao produce better clustering results as compared to Jaccard by reducing arbitrary decisions. Unbiased Ellenberg-NM substantially decreases number of arbitrary decisions as compared to Unbiased Ellenberg and Information Loss for non-binary features producing significantly better clustering results[19].	Proposed Jaccard-NM measure and Russell & Rao measure perform better than Jaccard measure for binary features, while for non-binary features; the proposed Unbiased Ellenberg-NM measure produces results which are closer to the decomposition [19].
5	Object Oriented Software Clustering[20]	Stability Result on K-Means is 76.15 %	They run their execution time experiments on an Intel i7 Processor at 2.8GHz and 8 GB of RAM. Analyzed version of GEF software graph has 664 vertices and 2849 edges. As it is stated before fast community algorithm has low computational complexity when compared to other clustering algorithms.
6	Szeged InfoPólus Cluster[21]		Automatic collection of the metrics and the calculation of the answers to the questions.
7	Hyper-heuristic based Multi-objective Genetic Algorithm[24]	mtunis Mean 2.310 STD 0.011	The comparison is based on three main objectives MQ, intra-edges and inter-edges; along with the number of evaluations. In all the six test problems the MHypGA produced high quality solutions with a computational time of nearly one-twentieth of the time expended by the Two-Archieve Multi-objective Evolutionary algorithm [24].

6. Conclusion and Future Work

According to this the opinion of Statistics create public is banner which has been addressed as facts warehousing, details mining and information systems. It has been old-created wind distressing data draught chief full force the draught of miserly of analyses and mosey it courage appropriately impact on decisions made on the basis of these results. An effort to lend mixture correctness by pre-clustering did not succeed. This method, irrationality put a damage on incarcerated clusters alien invest in the scenes sets were completely in accordance with error rates within the same clusters on the test sets. This improvement

could perchance be second-hand to venture relevance confidence levels for predictions. The wide-ranging and the routine establishment stroll the software utilization has to angle is the safeguarding safe keeping of industrial software systems. Two of the bimbo premises for the uppity exhortation of preservation are the rudimentary complication of associate software systems lapse are large, complex, inconsistent and integrated. The unspecified prove behind the top phenomena is for the purpose of alternative extent and level of arrangements. Ancient software encrypts into subordinate, just about bendable subsystems nub aid the process of understanding it significantly. Surrogate algorithms construct different decompositions. Note, it is banderole to have methods divagate assay the ambience of such automatic decompositions.

Based on our study we suggest some future suggestions which are following:

1) More experimental comparisons will be conducted on larger datasets, with different clustering algorithms.

2) Further interactions can be introduced to provide feedbacks to expert's decision (i.e., to assist the expert with future labeling efforts).
3) To investigate the potential of using clustering to identify noise in the dataset, and eventually evaluate the quality of software metrics collected.
4) Coupling and Cohesion will be better for reengineering.
5) Optimization techniques can be used for optimizing the defect analysis.

References

[1]. L. Guo, B. Cukic, and H. Singh, "Predicting fault prone modules by the Dempster–Shafer belief networks," in Proc. 18th Int. Conf. Automated Softw. Eng., Montreal, QC, Canada, Oct. 2003, pp. 249–252.
[2]. K. E. Imam, S. Benlarbi, N. Goel, and S. N. Rai, "Comparing case based reasoning classifiers for predicting high-risk software components,"J. Syst. Softw., vol. 55, no. 3, pp. 301–320, Jan. 2001.
[3]. T. M. Khoshgoftaar and N. Seliya, "Comparative assessment of software quality classification techniques: An empirical case study," Empir. Softw. Eng. J., vol. 9, no. 3, pp. 229–257, Sep. 2004.
[4]. Dishek Mankad," Risks Management in Software Engineering", International Journal of Advanced Computer Research (IJACR) Volume-2 Number-4 Issue-6 December-2012.
[5]. T. M. Khoshgoftaar, S. Zhong, and V. Joshi, "Noise elimination with ensemble-classifier filtering for software quality estimation," Intell. Data Anal., vol. 9, no. 1, pp. 3–27, 2005.
[6]. H. Zeng, X. Wang, Z. Chen, H. Lu, and W. Ma, "CBC: Clustering based text classification using minimal labeled data," in Proc. IEEE Int. Conf. Data Mining, Melbourne, FL, 2003, pp. 443–450.
[7]. Saifi Bawahir, Mohsin Sheikh," Data and Cost handling Techniques for Software Quality Prediction through Clustering", International Journal of Advanced Computer Research (IJACR), Volume-2 Number-4 Issue-6 December-2012.
[8]. S. Zhong, T. M. Khoshgoftaar, and N. Seliya, "Analyzing software measurement data with clustering techniques," IEEE Intell. Syst., vol. 19, no. 2, pp. 20–27, Mar./Apr. 2004.
[9]. Rashmi N, Suma V," Defect Detection Efficiency: A Combined approach", International Journal of Advanced Computer Research (IJACR), Volume-3, Number-3 Issue-11, September-2013.
[10]. W. Zhang, H. Zhao, and H. Mei. A Propositional Logic-Based Method for Verification of Feature Models. In Proceedings of the 6th International Conference on Formal Engineering Methods (ICFEM 2004), volume 3308 of Lecture Notes in Computer Science, pages 115–130, Seattle, WA, USA, November 2004. Springer-Verlag.
[11]. Briand, L., Wust, J., Daly, J., Victor, P.D., 2000. Exploring the relationships between design measures and software quality in object-oriented systems. J. Syst. Software,51(3):245-273.
[12]. Reformat, M., Pedrycz, W., Pizzi, N.J., 2003. Software quality analysis with the use of computational intelligence. Inf. Software Technol., 45(7):405-417.
[13]. Kanmani, S., Uthariaraj, V.R., Sankaranarayanan, V., 2007.Object-oriented software fault prediction using neural networks. Inf. Software Technol., 49(5):483-492.
[14]. Pragati Shrivastava, Hitesh Gupt," A Review of Density-Based clustering in Spatial Data", International Journal of Advanced Computer Research (IJACR) Volume-2 Number-3 Issue-5 September-2012.
[15]. Mark Shtern and VassiliosTzerpos," Methods for Selecting and Improving Software Clustering Algorithms", IEEE 2009.
[16]. Ramandeep S. Sidhu, Sunil Khullar, Parvinder S. Sandhu, R. P. S. Bedi, KiranbirKaur, "A Subtractive Clustering Based Approach for Early Prediction of Fault Proneness in Software Modules", World Academy of Science, Engineering and Technology,2010.
[17]. Mark Shtern and VassiliosTzerpos "On the Comparability of Software Clustering Algorithms" Proceedings of the 18th IEEE International Conference on Program Comprehension, Braga, Minho, June-July 2010, pp. 64-67.
[18]. Jin-Cherng Lin; Han-Yuan Tzeng, "Applying Particle Swarm Optimization to estimate software effort by multiple factors software project clustering," Computer Symposium (ICS), 2010 International , vol., no., pp.1039,1044, 16-18 Dec. 2010.
[19]. Naseem, R.; Maqbool, O.; Muhammad, S., "Improved Similarity Measures for Software Clustering," Software Maintenance and Reengineering (CSMR), 2011 15th European Conference on , vol., no., pp.45,54, 1-4 March 2011.
[20]. Erdemir, U.; Tekin, U.; Buzluca, F., "Object Oriented Software Clustering Based on Community Structure," Software Engineering Conference (APSEC), 2011 18th Asia Pacific , vol., no., pp.315,321, 5-8 Dec. 2011.
[21]. Beszédes, A.; Schrettner, L.; Gyimóthy, T., "Development of a Unified Software Quality Platform in the Szeged InfoPólus Cluster," Software Maintenance and Reengineering (CSMR), 2012 16th European Conference on , vol., no., pp.495,498, 27-30 March 2012.
[22]. Deepak Gupta, Vinay Kr. Goyal and Harish Mittal, "Analysis of Clustering Techniques for Software QualityPrediction", 2012 Second International Conference on Advanced Computing & Communication Technologies.
[23]. Dhiman, P.; Manish, M.; Chawla, R., "A Clustered Approach to Analyze the Software Quality Using Software Defects," Advanced Computing & Communication Technologies (ACCT), 2012 Second International Conference on , vol., no., pp.36,40, 7-8 Jan. 2012.
[24]. Kumari, A.C.; Srinivas, K.; Gupta, M.P.,"Software module clustering using a hyper-heuristic based multi-objective genetic algorithm,"Advance Computing Conference (IACC), 2013 IEEE 3rd International, vol., no., pp.813, 818, 22-23 Feb. 2013.
[25]. Beck, F.; Diehl, S., "Evaluating the Impact of Software Evolution on Software Clustering," Reverse Engineering (WCRE), 2010 17th Working Conference on, vol., no., pp.99, 108, 13-16 Oct. 2010.
[26]. Tianmei Zhu; Yijian Wu; Xin Peng; Zhenchang Xing; Wenyun Zhao, "Monitoring Software Quality Evolution by Analyzing Deviation Trends of Modularity Views," Reverse Engineering (WCRE), 2011 18th Working Conference on , vol., no., pp.229,238, 17-20 Oct. 2011.
[27]. Wu Ren, "Mining Association Rules to Facilitate Structural Recovery," Computer Software and Applications Conference Workshops (COMPSACW), 2012 IEEE 36th Annual , vol., no., pp.272,277, 16-20 July 2012.