What Are the Software Engineering Problems? Are We With The Right Approach?

M Uma Devi
Assist. Prof., Dept. Of Computer Science
Sri Adi Chunchana Giri Women's College
Cumbum, India

J Sandeep
Research Scholar, Dept. Of Computer Application
Bharathiar University
Coimbatore, India

ABSTRACT
The recent development of the technologies in the day-today life has made the people committed with more and more software's. The software engineering is centered on a key attribute the Software Reliability, which is defined as the probability of failure free executions. In this competing world of technologies, applications with newer and better solutions only survive. Developing this modern software with right approach still remains as one of the main software crisis. Thus it makes it crucial to follow the right approaches with a good knowledge of the software engineering problems, their approaches and expected results. In this paper the software engineering and their various Approaches are discussed. The categorization of software engineering problems and their possible results are made for addressing the issue raised due to the wideness of software engineering. A view on various validation methods exist in software engineering are discussed for making the software more reliable. The paper explores promising research areas in software engineering for exclusive upcoming researchers.

General Terms
Software Engineering Problems and various approaches used.

Keywords
Software engineering; result interpretation; validation; reliability.

1. INTRODUCTION
We Software Engineering [1] is the technological and managerial discipline which is concerned with systematic production and maintenance. In software engineering the production and modifications are made with dependence with cost and time for the engineering. It serves as the basis for modern scientific world with the intention of investigation and problem solving. Making the project achieve success requires many factors to be considered right simultaneously. Software project always aims to achieve its highest profitability. A petty flaw in planning or management can lead to destructive results. Building effective software involves in-depth investigations into various aspects. It is important to develop software with innovative characteristics that make it different from other things that human beings build. For that, one should analyze the complexities on software engineering, however, may result from only one slight problem. It is obvious how important to analyze the problems when researching on software engineering. Software engineering concepts are often characterized by a number of assumptions. The observations, from both empirical and analytic studies, show that the predictions made by the concept tend to be too optimistic and should reformulate to control software engineering problems [2]. The most prominent limitations of the concepts are as the software behavior changes because the software code changes during the testing leads to complexities in real time. This kind of measurements reflects on the wideness of areas to improve the system quality. It depends on different factors as system and user. So, there is some importance to develop such software has become an extremely challenging job not only because of inherent complexity, but also mainly concern with the number of factors that categorized as adaptability, modularity, profitability and users' expectations to perform problem solving processes. The main contribution of this paper is in-depth evaluation of wideness of this area and importance of categorization. This has lead to a number of remarkable observations concerning the wideness and the importance of categorization. The basic inspiring task of software engineering exposed in fig1.

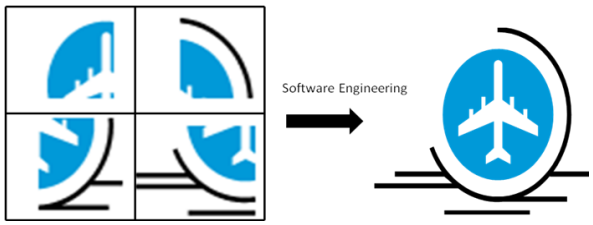

Fig.1. Task of Software Engineering

The remaining sections of this paper are organized as follows. Section 2, we introduce shows the different types of software engineering problems that could possibly appeal to be considered the right approach. Section 3 summarizes the results and interpretations used to express the proposed approach. In Section 4 an approach to get results on validation are discussed. Finally the paper is concluded from the studies made and discussed.

1.1. Software engineering and its different types of research Questions
Software has developed to satisfy some specific goal. Software measurement is based on numeric value derivations for an attribute of a software product. Here, identified measured value is mapped from the empirical world to the formal world. [3] Defines design complexity needs an iterative process of identifying design problems and solving design problems. It must be decomposed on projects in a number of dimensions to get its consistency. The problems may arise in a number of ways as software development methods, analyzing methods, how to evaluate software, integration of software with the whole classes of systems, about the feasibility and how to implement the developed product to the systems.

1.2. Problems at the Development Phase

The software engineering research question begins from the development phase. Here the first question arises is what product to develop? What is the better way to develop? There are several entities which affect the development of the project.A software system progresses towards several state changes during its development [4]. Thus the method requires better understanding of those entities. We have to assign the requirements with suitable structures to ensure flexibility and robustness [5]. Stepwise refinement, starting from a higher level description of what the program is intended to do and then gives various levels of pseudo-code until the low-level code is in place [6]. A project development concern must have the concern factors in fig. 2.

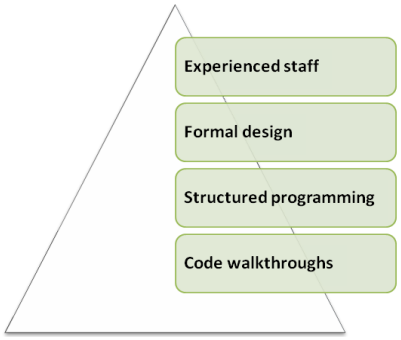

Fig. 2. Project Development Concerns

The other problem can be that the communication gap between the end-users and the software developers is increased which leads tedious time taken to solve small bug. Some methods implemented in software confusing like reliability growth. So there is much necessity of research on how these technologies really work. A good software development involves following things as shown in fig. 2

1.3. Problems arise at analysis phase

The Quality or the Correctness of the software result always remains a question in the software engineering. The use-case diagram shows the important parameter of software quality in fig. 3. As it is known that the knowledge gathering differs from people to people. To satisfy user satisfaction against quality, it needs to be analyzed much.

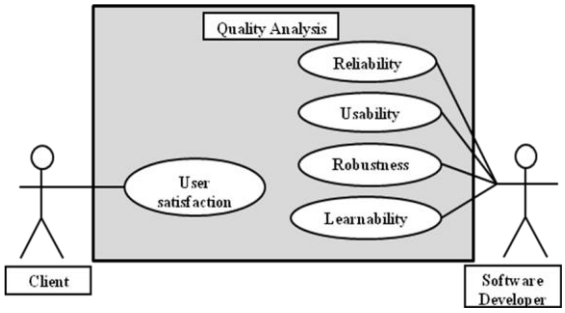

Fig. 3. Use-Case Diagram: Software Quality Parameters

Nowadays software is analyzed in a number of aspects in order to generalize the facts to everyone. First of all, data analysis [7] is too difficult to find out complete bugs. They had made 21000 test cases in automatic airplane landing system. It is difficult to make quantitative predictions, finding time-consumption and test hypothesis. Sheppard and Kruesi [SHEPPARD] examined the performance of programmers in constructing programs from various specification formats. An analysis of the error data revealed that the major source of difficulty was related to the control flow. We have analyzed the software quality parameters against user's satisfaction which is exposed using a use - case diagram in fig 3.

1.4. Problems at Evaluation Phase

In testing, test case is a way of doing software activities. It is a set of procedure in order to find out defects. A different test - cases make different efficient rate.

Fig.4. Cause – Effect diagram: Project Evaluation

The cause-effect diagram shows how the project evaluation is done with the step by step activities in testing as shown in fig.4. It is helping to identify defective in different modules by point out highly complex and unstructured code. The ability of the model is evaluated to find defect-free modules.

1.5. Integration with the whole classes of systems

A complex work decomposed into a number of modules in the project. There are no communication gap and navigational problems when using with interfaces between the numbers of modules which leads to the successful running project. Integration process needs support of both hardware and software.

1.6. Feasibility

Feasibility study in software engineering is to determine whether it is reasonably economically and technically feasible to develop the product. Feasibility is not only to solve the problem but to determine whether the problem is worth in solving. Thus the feasibility will include the verification of operational, technical and economically feasible.

1.7. Implementation of the system

Post release defects should be removable to get better usage of products. [22]The investigation of OO software product metrics is used to predict defects after software releases. This approach allows early identification of error-prone classes and direction of quality activities like testing and refactoring. It collects software internal and defect history metrics, from Version Control and Issue Tracker systems, and to integrate measures in a consolidated table, adequate for defect Prediction. The individual software and real-world concerns are shown in fig. 4.

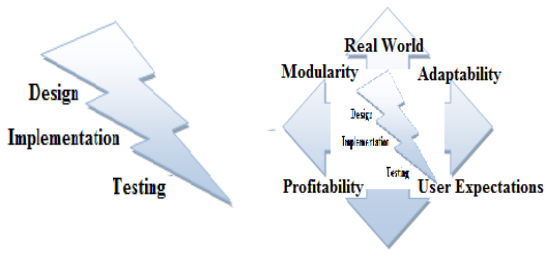

Fig.4 (a). Narrow Analysis (Individual software concerns) (b). Broad Analysis (Real World Concerns).

The narrow and broad controls of software engineering are exposed with the importance of categorization in fig. 4. Software engineers should implement the users' specification as input to get the expected output. A parameterized complexity in modeling [8] observes the behavior of already existing system input/output specifications of the to-be-modeled system.

2. SOFTWARE ENGINEERING RESULTS & INTERPRETATION

Getting results always direct the fulfillment of expectations. Training data will be the input and a result comes after the data flows through different approaches. The result even depends on the approach interpreted to show the results of complete work. Thus the results can be based on different approaches and among which few are discussed below.

2.1 Judgment based result

It is very useful to take a decision in further. Always a right decision making has been a step towards success. In this aspect, judgment based results sometimes good [9].

2.2 Prediction based result

Sometimes prediction based results produce irregular depends on system. For example, weather casting. The forecasting values checked with already predicted values [9]. The researchers believe they can further improve the prediction accuracy of these models by adding other important factors which affect the final software quality [10] [11]. Software engineering, like other engineering fields, needs to formalize, standardize, create uniformity and have certain level of predictable functionality [12]. Statistical debugging approaches [13] [14] identify software faults with probabilistic modeling of program predicates. It is the kind of assessment of codes with respect to software faults.

2.3 Empirical results

They[15] have identified empirical evaluations of commercial software systems in terms of size, customer usage and economic value.[16] empirically identified object oriented models with specific characteristics structural aspects, behavioral aspects, combination of structural and behavioral aspects.

2.4 Evolution based result

These kinds of result always produce the merits and demerits of our thoughts. [17] Presents the strength and weakness of different models by criteria-based evaluation method.

2.5 Estimated result

The different experiments conducted to evaluate the performance of our transliteration algorithm [18] and compared the performance against the statistical CRF-HMM system by Surya et.al [19].

2.6 Report generation

The number of data used for training to optimize the particular output. For that, the collection of statistical data was experimented for obtaining consistent result in the form of a report [20].

2.7 Specific solutions

The particular solutions obtains always depends with some models and its architecture. So the model [21] performs some statistical methods and produces the results with significance.

3. SOFTWARE ENGINEERING RESULT VALIDATIONS

A research is not only needs its results but also clean evidence or proofs that the generated output result is good. There are several methods used for validating the result which is discussed below.

3.1 Comparisons

The software results validation of this method is done based on the comparisons with similar existing or with the sample from the model. The results are normally analyzed for the satisfactory with rigorous derivations and proofs.

3.2 Experience

In this method, the results are validated by someone with his experience of similar work and results. Here the real examples are taken for the correctness and effectiveness of the developed model compare to the alternative ones. The experience based models compare the method and techniques of similar results. In this model statistical and on practice comparisons are made after the data.

3.3 Evaluation

One of the software results validating method is the evaluation method were stated criteria are developed and the results are validated as per the phenomena of interest. Example of this method is the use of the descriptive model where it adequately describes the phenomena of the interest and then the results in real time validated with this interested phenomena.

4. CONCLUSION

Software Engineering is the management of discipline concerned with systematic and advance approaches for production and maintenance to utilize the cost and time with minimal risk and maximum gains. In this paper from the studies and discussions it is concluded that software engineering will be benefited with the better understanding of various strategies and successful research approaches. Every research question varies with its type and same time it needs different research strategy to resolve. Software engineering projects need to select an in proper approach to obtain a result and validation strategy appropriate to the question.

5. ACKNOWLEDGMENTS

The authors are thankful of their parents and to the faculty members of the academic college and university for their support. The Authors wish to thank the reviewers and the editors for their valuable suggestions and comments that helped to improve the paper.

6. REFERENCES

[1] R. Fairly, "Software Engineering Concepts", McGraw-Hill, 2010.

[2] A.S. Kalaji, R.M. Hierons and S. Swift, "An Integrated Search-Based Approach for Automatic Testing from Extended Finite State Machine (EFSM) Models", Information and Software Technology, Vol. 53 Issue 12, Dec. 2011.

[3] A. Tang and A. Aleti, J Burge, and H V Vliet, "What makes software design effective? ", Design Studies Vol. 31 Nov. 2010.

[4] Fujitsu and I. Jacobson, "Essence – Kernel and Language for Software Engineering Methods", 13 Aug. 2012.

[5] M. Selvam and A. M. Natarajan, "Language model adaptation in Tamil language using cross-lingual latent semantic analysis with document aligned corpora", Current Science, Vol. 98, No. 7, 10 Apr. 2010.

[6] K. Rustan M. Leino and Kuat Yessenov, "Automated Stepwise Refinement of Heap-Manipulating Code", 2010.

[7] Y.K. Malaiya, Senior Member, IEEE, Michael Naixin Li, James M. Bieman, Senior Member, IEEE, and Rick Karcich, " Software Reliability Growth With Test Coverage", 420 IEEE Transactions On Reliability, Vol. 51, No. 4, Dec. 2002.

[8] I.V. Rooij and T. Wareham, "Parameterized in Cognitive Modeling: Foundations, Applications, Opportunities", the Computer Journals, Vol. 51, No. 3, 2008.

[9] S.H. Aljahdali and Khalid A. Buragga, "Employing four ANNs Paradigms for Software Reliability Prediction: an Analytical Study", ICGST-AIML Journal, ISSN: 1687-4846, Vol 8, Issue II, Sept. 2008.

[10] M. Chen, M.R. Lyu, and E. Wong, "Effect of Code Coverage on Software Reliability Measurement", IEEE Transactions on Reliability, vol. 50, no. 2, Jun 2001, pp. 165-170.

[11] Y.K. Malaiya, N. Li, J.M. Bieman, and R. Karcich, "Software Reliability Growth with Test Coverage", IEEE Transactions on Reliability, vol. 51, no. 4, Dec 2002, pp. 420-426

[12] B.K. Olorisade, "Informal Aggregation Technique for Software Engineering Experiments", IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 5, No 1, Sept. 2012.

[13] C. Liu, L. Fei, X. Yan, J. Han, and S. Midkiff, "Statistical Debugging: A Hypothesis Testing-based Approach," IEEE Transaction on Software Engineering, vol. 32, no. 10, Oct, 2006, pp. 831-848.

[14] A.X. Zheng, M.I. Jordan, B. Libit, M. Naik, and A. Aiken, "Statistical Debugging: Simultaneous Identification of Multiple Bugs," Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh, PA, 2006, pp. 1105-1112.

[15] T. Bhat, N. Nagappan, "Building Scalable Failure-proneness Models Using Complexity Metrics for Large Scale Software Systems", 2006.

[16] N. F. Schneidewind, "Analysis of object-oriented software reliability model development", Innovations System Software Engineering, 2009.

[17] R. Ramsin, "The Engineering of an Object-Oriented Software Development Methodology", Apr, 2006.

[18] S. Sethuramalingam, "Effective Query Translation Techniques For Cross-Language Information Retrieval", International Institute of Information Technology, Hyderabad, India, Sept. 2009.

[19] S. Ganesh, S. Harsha, P. Pingali., And V. Varma, "Statistical transliteration for cross language information retrieval using hmm alignment and surf", Proceedings of the 2nd workshop on Cross Lingual Information Access (CLIA) Addressing the Information Need of Multilingual Societies. ACL, 2008.

[20] A. Das, T. Saikh, T. Mondal, A. Ekbal, S. Bandyopadhyay, "English to Indian Languages Machine Transliteration System at NEWS 2010", Proceedings of the 2010 Named Entities Workshop, ACL 2010, pages71–75, July, 2010.

[21] N. Salman, "complexity Metrics AS Predictors of Maintainability and Integrability of Software Component", Journal of Arts and Sciences, 2006.

[22] Gabriel de Souza Pereira Moreira,, Roberto Pepato Mellado, Robson Luis Monteiro Junior, Adilson Marques da Cunha and Luiz Alberto Vieira Dias, "Predicting Post-Release Defects in OO Software usingProduct Metrics", Proceeding of IX Experimental Software Engineering Latin American Workshop,2012.