A Review on Software Maintenance Cost Evaluation

Reenu Rani, Amandeep Kaur Cheema

Dept. of CSE, guru nank dev university Amritsar (Pb.) India

Abstract- Software reuse and software maintenance are critical issue in the software reengineering. Software maintenance cost defines the time when to redevelop a software product also software define select those components among already available software for reuse. This paper evaluates the various techniques to evaluate the software maintenance cost or software reuse. Various static and dynamic metrics are used to find the whether or not to reuse the software component and also various metrics define the software maintenance cost.

Index terms: Software re-engineering, software reuse, software maintenance.

I. INTRODUCTION

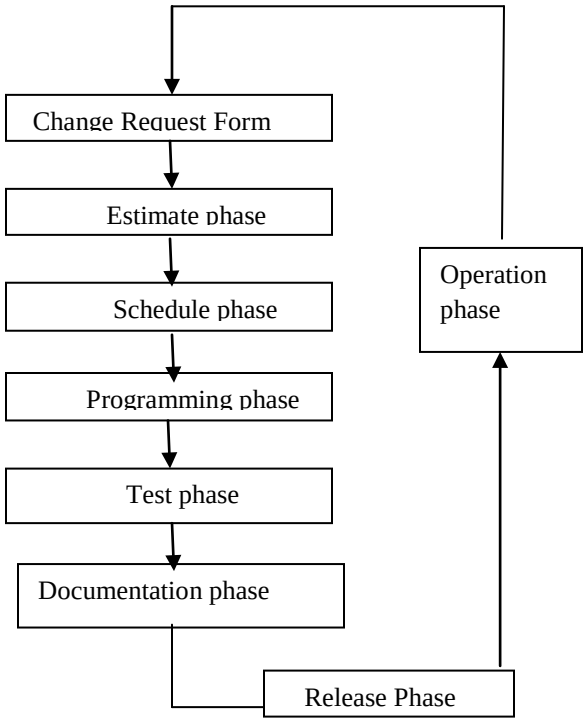
Software Engineering is a coated technology. Software Engineering implements strong engineering principles to provide economical software that will reliable and provides efficient performance on real machines. It is the study of engineering to the plan, improvement, and maintenance of software.

Software Development is the improvement of software program. It includes activities involved more than one person which depends totally on the size of the software. Each phase of the development is comprised of various activities of the individual.

There are different types of software's are implemented in software engineering. Software is set of lines of code that on execution provides the desired productivity. Data Structures helps the code to store the information and store them, manipulate them and finally convert them to the nearest set of output. Various programming languages are used during the software development. Due to the recent evolution of the computer's in all the fields software engineering provides a lot of features in the form of products, software's, systems and information systems. Software development have different phases in which software maintenance is last and important phase. Software maintenance is correction of errors in software and adds some functionality to make it adaptable after delivery of that software. Software maintenance is process that is used to adapt, organize, improve, prevent and maintain the performance of any software. In software development it is very important to think about software maintenance.

Purpose of software maintenance: In today's world, customer's requirements are changing day by day. Customers want that software which provides an accurate work. To make software adaptable with changing environment there is need of maintenance so software should be developed in this way so it can be maintain easily. Software maintenance makes a software correct, perfect, preventive and adaptable with changing environment

Software Maintenance This phase focuses on corrective the bugs and errors in the software after testing the software. This is the important phase of software development as the bugs are meant to be fixed within reasonable time with less cost. This includes enhancements also that come from the changing requirements from the customer.



Need of maintenance: Our environment is dynamic in nature. User requirements, economical and technical changes are taking place in our environment so there is need of modification to keep it adaptable with environment. To meets today's requirement maintenance is very important part. It is last phase of software development life cycle. All the software's will need maintenance over there lifetime.

- If there is serious error in software then there is need of software maintenance to keep it compatible.
- New business has different new processes for cooperation there is need of maintenance in existing system.
- For security vulnerability of software there is need of software maintenance.
- When hardware is changed then there is need of modification of software to make it compatible with that hardware so there is also need of maintenance

Maintenance cost and cost factors: Maintenance cost is greater than development cost. Software maintenance damages the product formation. It makes advance maintenance more complex. The software's which are developed with old languages and compilers needs more time and cost for maintenance.

Cost factors:

1. *Team stability:* If the same staff who has developed software are involves in software maintenance then maintenance cost can be reduced
2. *Contractual responsibility*: The developers of software may have no contractual responsibility for software maintenance so there is no motivation to plan for upcoming modify.
3. *Staff skills*: Maintenance personnel are frequently new and have restricted area familiarity.
4. *Program age and structure*: As programs age, their formation is corrupted and they be converted into harder to realize and modify.

Maintenance Cost evaluation: Changing software after post delivery is called software maintenance. Software maintenance cost is the cost required to maintain the software after initial deployment. Customer's needs are changing day by day. Software maintainability is important part for success because product does not wear out but the use of that product will get less. Cost will be high if the software is not having reusable components as it is the important key in changing the system to new changes.
Software maintenance cost include

- Corrective maintenance: Software with the best quality also encounters defects but the corrective maintenance helps to fix those defects. Cost required for bug fixing after initial deployment is known as corrective maintenance cost .It is generally 20% of all software maintenance cost.

- Adaptive maintenance: Due to the changes in the environment including CPU, Operating System, Business parameters, software meant to be changed and adopt the new changes in environment. Cost required to make software adaptable in all environments is known as adaptive maintenance cost. It is generally 25% of all software maintenance cost.

- Perfective maintenance: To improve and enhance the overall performance of the software after initial deployment is perfective maintenance .It is generally 5% of total software maintenance cost.

- Enhancements: Costs due to long-term new creations and innovations in software. It is generally 50% or more of total maintenance cost of that software.

The cost evaluation is one of the economical techniques which is used to choose an well-organized program from wide range of alternatives and to propose and execute that program. The principle of cost evaluation is do more with less budget. Cost evaluation identify most efficient approach of software.

Maintaining a software with maximum profit and minimum cost is known as optimal maintenance. Cost functions are depend on the reliability and maintainability individuality of the software find out the parameters of significance to decrease. Parameters measured are – the cost of collapse, – the cost per time unit of "downtime", – the cost (per time unit) of corrective maintenance, – the cost per time unit of preventive maintenance an – the cost of repairable system replacement. Maintenance cost increases above time to time. A software which has been delivered is exclusive to change. Maintenance costs increase over time and as the

system evolves Reasons: Maintenance changes, degrades the original system structure. Aging software results in high support costs.

II. RELATED WORK

Imai et al. [1] said that, code cloning is frequently evaluated by most of the software companies. The source code which is used by more than one project can increase the maintenance cost. When software functionalities are modified and some changes are required due to changing environment then source code cloning difficulty for that software project. This paper describes the maintenance cost evaluation which evaluates the impact of reuse to maintenance. The main motive of this paper is to measure the functional redundancy. It can be measured in two steps: first to make clusters of similar functions which are used in that source code and second to make an array weighted FR-tree according to that clusters. After that, functional redundancy can be measured by weight of each node in FR-tree.

Srivastava and Biplav [2] describe that, Software Engineering is all about combining the components with each other and then assembled it into Software. These components are further comes up with combination of sub-modules. One of the main tasks in managing software project involves having a track of development and managing the individual components. Though tools are there to track the component development and maintenance but still the main and important hurdle it to evaluate trade-offs manually. Software engineering is one of the fast growing fields and new trends are there like Web Services, EJB's, so there is a need of an automated solution to help developers in creating complex applications. This paper proposed an automated decision support framework that will guide the users to make cost-effective decisions. The key approach of this paper is to develop a model of the software using the automated planning techniques to create alternative plans to develop and maintain software's taking into considerations of the developer's efforts and objectives.

Nagappan and Ball [3] have proposed that Software development is complex task in which there is need of complete knowledge about the architecture of that particular software. This paper describes, how the software dependencies and churn measures are prone to post release failures. In this paper, author analyzes the window server 2003 operating server and examines the relationship between software dependencies and churns measures and also investigates their ability to proneness post release failures. Churn measures are the measurement of code which is changing with in software unit with the analysis this paper conclude that software dependencies and churn measures arethe efficient predictors for post release failures and failure proneness. It also concludes that software quality can also be measured with this prediction and post release failure proneness is also economically useful.

Wolf et al. [4] described that Communication is the way through which ideas, thoughts and knowledge can be share in work groups. Communication and coordination plays an important role for software quslity. This paper describe the relationship between successful coordination outcome and communication structure using the data from the project of IBM (Jazz ^{TM} project) . This paper introduces two research question and methodology in the study of integration and communication structure in Jazz project. Author conceptualizes the coordination outcome with the success or failure of the project and study communication structure of team with social network measures. The paper concludes that coordination outcome and communication structure are the efficient predictors for failure in future. Problems in communication will lead to coordination and integration failures.

Cataldo et al. [5] describe past research has shown that customer reported software faults are the result of violated dependencies that are not recognized in software developing by software developers. There are three types of dependencies which are included in this paper. The main objective of this paper is to find the performance of the dependencies and relate it to customer reported defects. Analysis of this research is based on two different projects of different companies. Research related to this topic shows that logical, syntactic and work dependencies all prone to failure. Prior research has shown that syntactic dependencies were more prone to failure on the other hand this research shows that logical dependencies and work dependencies are more prone to failure. The result of this paper suggests that rearchitecting guided by the network structure of logical dependencies are useful for reducing defects.

Mader, Patrick, and Alexander Egyed[6], have proposed that Effects of requirements traceability regularly helps the developer's to have a track of the software changes and makes the software maintenance support better. Though it is

quite popular approach, but still there is no publication about the benefits of the requirements traceability. It is quite important to learn that whether introducing requirements traceability can
make the development tasks better. This paper conducted an experiment using 52 developer's performing real time maintenance on the third party software, half of the tasks with traceability and half of them without traceability. The results shows that the developers with traceability performs 21% faster on a task and provides 60% more correct solution suggesting that traceability not only saves money but provides better maintenance. Using the initial costs setup for our evaluated traceability featured systems we will implement the same to improve the maintenance support quality.

Langelier et al. [7], describe that evaluating the software quality and understanding the events leading its evolution to anomalies are two key steps in reducing the cost in software maintenance. Evaluation of the large quantity code over various versions is the most time consuming and practised in general. To answer this, this paper described that a semi-automatic visualization framework to investigate programs having thousands of classes, over dozens of versions at much faster speed. Programs and their associated quality parameters for each version are represented graphically independently. The animations between these representations create visual quality of consistence associated with the quality belonging to the various software versions. Exploring these qualities parameters can reduces the difference between various views of software. This allows experts to use their analysis skills to investigate all quality aspects of software evolution.

Ghavami et al. [8], have proposed that a probalistic model to attain cost effective maintenance strategies. Customer's needs changes day by day so the maintainability is very important part for all software's. Maintenance increase the life time of any product and increase the meantime among failures, how easily software can be modified during bug fixes or other modifications is generally known as maintenance. Now days, maintenance cost effectiveness is important because of budget constraints of each software industry. Cost effective and profitable product will be more reliable. This paper describes that based on reliability indices mean duration and state probability are computed using Monte Carlo simulations and numerical examples; cost analysis is also performed with computation of all related cost like maintenance, failure cost based on reliability.

Liu et al. [9], have proposed novel maintenance strategy model, which focuses on the task of maintenance workers and the work provision method. This model can be used to estimate product reliability in different maintenance circumstances. One of the maintainability parameter is frequency of emergency state, which plays the crucial role in enforcing hierarchical planned maintenance. With reduction of hardware and software cost, total cost can be reduced. In this paper, the viability of this method can be revealed by a plain example. This assessment and model have been implemented as a software tool for dependability proposes.

Mitsuhiro et al. [10], have proposed that, product feature evaluation method in development process with the change of software metrics. It is broadly known that in software development companies, software evolution makes the product configuration difficult and it makes the product evolution more complex. This paper used huge size telecommunication software fixed in the radio network system which is described by C/C++ language. As the outcome, this
paper prove that betweenness , centralization and average distance of complex network metrics have great impacts on fault densities in software evolutions process.

III. CONCLUSION AND FUTURE WORK

This paper has evaluated various techniques for software re-engineering. Software quality metrics are used to find the software maintenance cost and also for whether the software is reusable or not. It has been found that the software maintenance plays a significant role to build the software product again called new version of that software. In near future we will use some reusable components and evaluate their reusable cost to find the best one; based upon certain quality metrics.

REFERENCES

[1] Imai, Takeo, Yoshio Kataoka, and Tetsuji Fukaya. "Evaluating software maintenance cost using functional redundancy metrics." Computer Software and Applications Conference, 2002. COMPSAC 2002. Proceedings. 26th Annual International. IEEE, 2002.

[2] Srivastava, Biplav. "A decision-support framework for component reuse and maintenance in software project management." Software Maintenance and Reengineering,

2004. CSMR 2004. Proceedings. Eighth European Conference on. IEEE, 2004.

[3] Nachiappan Nagappan, Thomas Ball, "Using Software Dependencies and Churn Metrics to Predict Field Failures: An Empirical Case Study", First International Symposium on Empirical Software Engineering and Measurement, 2007 IEEE.

[4] Wolf, Timo, Adrian Schroter, Daniela Damian, and Thanh Nguyen. "Predicting build failures using social network analysis on developer communication." In Proceedings of the 31st International Conference on Software Engineering, pp. 1-11. IEEE Computer Society, 2009.

[5] Cataldo, Marcelo, Audris Mockus, Jeffrey A. Roberts, and James D. Herbsleb. "Software dependencies, work dependencies, and their impact on failures." Software Engineering, IEEE Transactions on 35, no. 6 (2009): 864-878.

[6] Mader, Patrick, and Alexander Egyed. "Assessing the effect of requirements traceability for software maintenance." In Software Maintenance (ICSM), 2012 28th IEEE International Conference on, pp. 171-180. IEEE, 2012.

[7] Langelier, Guillaume, Houari Sahraoui, and Pierre Poulin. "Exploring the evolution of software quality with animated visualization." Visual Languages and Human-Centric Computing, 2008. VL/HCC 2008. IEEE Symposium on. IEEE, 2008..

[8] Ghavami, Mohsen, and Mladen Kezunovic. "Probabilistic evaluation of the effect of maintenance parameters on reliability and cost." Probabilistic Methods Applied to Power Systems (PMAPS), 2010 IEEE 11th International Conference on. IEEE, 2010.

[9] Liu, Bin, and Zhengguo Xu. "A new cost evaluation model of predictive maintenance for dynamic systems with hidden degradation." Chinese Automation Congress (CAC), 2013. IEEE, 2013.