International Journal of Innovation and Scientific Research
ISSN 2351-8014 Vol. 22 No. 1 Apr. 2016, pp. 1-7

http://www.ijisr.issr-journals.org/

Enhancement in Task-Oriented Software Maintenance Model by Using Requirement and Design Reuse Repository

Abdul Hammad[1], Akmal Rehan[2], Ahsan Latif[3], and Nayyar Iqbal[4]

Department of Computer Science, University of Agriculture, Faisalabad, Punjab, Pakistan

ABSTRACT: The purpose of this research is to make comparison of different software development models as well as software maintenance process models for the sake of presenting an enhanced maintenance model. The comparison is made on the basis of different criteria, such as functionality, usability, reusability, performance and availability. Software is developed to help the user of the system so that he/she can work more effectively and efficiently. In software development following phases are considered i.e. requirement gathering phase, requirement analysis phase, design phase, coding, and testing. The maintenance in working software is applied on the basis of following criteria: editing in software module, adding new features, removing useless features, and for enhancing functionality of complete software, so that the software product can cope with current working demands of market. To overcome the problems occurs in current working software there is need to make maintenance procedure more effective and perfect. The current study has been based on enhancement of software maintenance process. "Enhancement in Task-Oriented Software Maintenance Model by Using Requirement and Design Ruse Repository" is proposed in this paper, which is based on "Enhancement in Task-Oriented Model". Through which it would become easy to proceed maintenance procedure in a controlled, cost effective and in a time saving way. Different software houses have been studied for the sake of making enhancement in the software maintenance model. This proposed model has lot of benefits over previous maintenance model. Moreover this study increases the customer confidence over maintainer and also uses as helping hand for increasing developer's confidence over customer or user.

KEYWORDS: Requirement Gathering, Requirement Analysis, Cost Effective, Controlled and Time Saving, Reusability, and Functionality.

1 INTRODUCTION

Software quality indicators are divided into many types, Software Maintainability is also one of those most important quality indicators. Software quality can be accurately measured by software maintainability. It is very hard to find comprehensible and understandable view on all those factors that have significant effect on the quantification of testability of software, which is also a software quality indicator.

Software maintenance can be expressed as "The act of modifying already working part or module of software after handed over to customers for the sake of correcting errors, adding new features, removing old or obsolete features, enhancing performance and to change its foreground or background". And Software maintainability is defined as the ease of "modifying already working part or module of software after handed over to customers for the sake of correcting errors, adding new features, removing old or obsolete features, enhancing performance and to change its foreground or background". Software quality consists of many types of features; software maintainability is one of them. It is studied that about 60 to 70 percent expenditures are spent on adopting, correcting and maintaining existing software product. Maintainability quantification assists to monitor the efforts of maintenance at design stage. The Assessment of software design stage in software maintainability is more helpful for developing and maintaining the cost of software more effectively. Software maintenance process requires more effort to put than any other activity in software development process.

Corresponding Author: Abdul Hammad

Assessment of maintainability at early stages, completely at design phase is very helpful for designers to make advancement in their designs before the starting of coding process. Consequently, it significantly minimizes the rework efforts throughout and after implementation of software. This thing provides facility for making betterment in project planning, designing and implementation process. After the stating of implementation phase, applying changes in design phase in order to maintain the software will be very costly for developers. Whereas, quantifying maintainability before starting in the development process may considerably decreases the overall development time and cost. It's sorrowful for me to say that, a number of the software organizations are not just fail to produce quality software products but also some of them are not familiar with software quality attributes.

After the above discussion our finding is that the maintainability is a quality factor that tries to guess how much sources and efforts will be required for maintaining the software [1].

Maintenance can be regarded as a process performed during Software Development Life Cycle (SDLC) process of software but maintainability is one of the most important quality attribute. Better maintainability leads to reduced maintenance efforts and reduced cost and time. In hardware engineering, "maintenance" means keeping something in its original state. Software Maintenance means: Fixing design flaws (Corrective & Preventive maintenance), Modification for new uses (Adaptive maintenance), Improving performance (Perfective maintenance), arguably, and "maintenance" is the wrong word for any of this Software "maintenance" is actually post-deployment design [2].

2 REVIEW OF LITERATURE

The researcher defined that software maintenance process includes a number of activities. These activities are different for each model. If Quickly Modify Model is under consideration then these maintenance activities will be identify Problem and Correct errors. If there is Boehm Model then these maintenance activities will be Management Policy decision, implementation change, software delivery, and assess. Hence, each model has its own stages for maintenance. And each model has its own merits and demerits. Each model is selected according to its reputation. Code focused maintenance model is only focused on performance by taking into consideration the source code of the written program [3].

The researcher explained that the Task Oriented Software Maintenance Model defines how different tasks in a chain have to be performed in the environment of the maintenance objectives and various constraints of maintenance project. These three features understand the software module, modification of the software, and revalidation of the software under study. The discussed model consists of these features. It additionally shows that the software maintenance procedure consists of a group of systematic and well defined tasks. This described software maintenance process examined the documents, existing source code, and the modification requests which has been made by customer as its input [4].

The researcher described that the main goal of software engineering is to develop software product with high reliability, good quality and having low cost. So, the basic concept of this paper is to minimize the cost of software development phase during the stage of maintenance. SRGMs which is abbreviation of Software Reliability Growth Models is described in this research, this proposed new method of software reliability growth models (SRGMs) is based on Non-Homogeneous Poisson Process (NHPP) model. This model is presented for describing reliability growth during the development test stage. The results of this research provide lower cost, shorter schedule and good quality [5].

The researcher stated that there are different types of critical factors which affect the maintenance of the software. In maintainability system undergoes many types of changes. The impact of these changes is on Interface, Components, and Features. Object Oriented Programming (OOP) is one of the mostly used programming paradigms. Object oriented software maintainability has been studied through years and several researches proposed a high number of metrics to measure it. But there is no standardized method has been adopted to measure the software maintainability [6].

The researcher presented that IDE which is abbreviation of Integrated Development Environment is the most basic tool which is used by software maintainers currently to perform system maintenance activities. Prototype IDE was developed and tested that presented behavioral design information in the form of graph. While using an IDE based on behavioral view two experimental maintenance tasks are shown in this study. In first maintenance task, users were able to find the appropriate piece of code and modify that appropriate piece of code more easily and quickly. Using an IDE which is based on a behavioral view of the software as compared to conventional and traditional structured oriented model, it is studied that IDE provided more good results [7].

The researcher explained that millions of lines of code could have to be faced, depending on the system involved for fixing software bugs. So, the first task is to investigate the smaller lines of code where changes have to be applied. After that a deep study is used for checking the suitability of changes that we want to make in software. At the end, we have to identify

the exact place where we want to make changes. It is also studied that the effect of changes might affect the other part of software. However, in large systems you have to familiar with specific lines of code and question for according which changes have to be made [8].

The researcher explained the method which used to tackle with problems of design of product and process of reuse. The idea of modularization is introduced in this research. This idea is based on configuration technology. An information model is introduced in study which is used for the elaboration of design process. This research describes the concept of DPM which is the short form of design process module. It is three layer architecture. These three layers are module, activity and parameter. A DPM structure tree is created for the accessibility of hierarchical description. At the end, on the basis of DPM, an implementation framework is constructed for supporting reuse and reconfiguration of design process [9].

The researcher stated that software maintenance is combination of correcting revealed faults, adapting the system in a changed environment. Maintenance activities also applied for improving the system's performance and reliability. These mentioned activities result in the modifications of software system that are distributed to customers. Post release changes to the Microsoft system are presented and applied after releasing the new version of Windows by Microsoft Windows Serviceability team. Extensive testing of all fixes is applied which is the best method to avoid negative effect of regressions. For fixing calculated or estimated problems we must have to distribute our limited resources in most effective and efficient way. So, a tool is required that can predict the amount of risk for each and every fix. BCT which is short form of Binary Change Tracer tool is introduced to fulfill this need [10].

The researcher described Constructive Cost Model (COCOMO) which is very famous model for estimating cost. This model is an abbreviation of constructive cost model. This model specially designed for industry. This model was basically designed for cost estimation of a software process. After that, a software process improvement in Sudanese organizations was studied. Everybody knows that software is an intangible product and it plays vital role in our lives. Some software houses were visited in Sudan to understand their working procedure. And interviews were conducted with their expert persons to make estimation about the existing flaws. So that, these flaws could be removed [11].

The researcher explained that for reducing cost of development and for improving software quality software reuse is a solid plan in early days. But in these days, reusing software modules has been encouraged by component based software development method. Traceability is a main factor for reusability of software modules or components. Component composition is described from architecture of software modules. The links between these components should be defined formally. The component model that is constructed by reusing different modules has rigorous semantic. Component based software development means constructing software by assembling pre-designed, pre-developed, configurable, and independent software components. It is essential to brought up systematic methodology to guide the CBSD (component based software development) process [12].

The researcher stated the definition of maintenance in which it is defined that, to make changes in software after handing over it to customer. There is another name of maintainability which is repair-ability. Repairability means to correct faults in software. It is also called corrective maintenance. Maintenance has following types, corrective, adaptive, perfective and preventive. In short, the correct structure of modules of the software product very necessary for performing the repairance of the software. However, if repairability decreases then the reliability increases. If there are complexities and irregularities in source code, then it will be difficult for programmers to read and maintain the code. Maintenance cost is measured through calculating the total number of errors repaired [13].

The researcher defined that the estimation of cost in maintenance phase which is compulsory to guess the reliability. This is also essential for improving the productivity, planning of project, adaptability and controlling of the software. Precise and accurate estimation of cost makes better understanding between the user and customer. Software Maintenance phase took part in performing important role in the phases of different software development models. These popular software development models are Agile-development, Iterative development, and Component Based Software Development (CBSD) model. Software development process is turned into engineering process with the introduction of Component Based Software Development model [14].

The Researcher quantified that software has become very essential part of computer systems. Large amount money is spent on the software development phase due to human effort and maintenance process. There are many variables which such as, human, technical, environmental and political factors which affect the cost of developing software. There are many cost estimation models which were discussed already, involves Software Life Cycle Management which is known as SLIM model. SEER-SEM model which is abbreviated as Software Evaluation and Estimation of Resources-Software Estimating Model and Constructive Cost Model (COCOMO) model. These models are also in use now a days because of their efficient results. The model which is under discussion here is Software Maintenance Cost Estimation (SMCE) [15].

The researcher described that in the software development process a paradigm shift has been taken place during the past decade. Improvement in the internet technology has provided simplicity in the software development process. This internet technology is distributed everywhere on different geographical locations. This Open Source Software system is served as main components of discussed critical infrastructures in the society of technology which is still growing now. The active involvement of users is necessary for reporting bugs, request for adding new features and improvements in existing feature. These active users are spread across different geographical region [16].

The researcher described that Test requirement models takes part in playing a significant role in model-driven testing for net-centric and SOA based systems. These modeling requirements can explain multiple types of test requirement of SOA based systems. This type of lacked functionalities can cause negative effects on test sufficiency and increase test burden. In this research paper, a task scenario oriented test requirements modeling method is proposed which is named as TabTre. This proposed method is used to describe functional requirements, performance requirements and reliability test requirements in a combined way [17].

3 DEVELOPED MODEL (TASK ORIENTED SOFTWARE MAINTENANCE MODEL WITH REQUIREMENT AND DESIGN REUSE REPOSITORY)

The presented model is "Enhancement in Task-Oriented Software Maintenance Model by Using Requirement and Design Reuse Repository". Reusing requirement and reusing design modules are added in it which was not mentioned in previous studied model "Task-Oriented Software Maintenance Model". In this developed model authors made the modifications with respect to requirement reuse and design reuse. The reuse modules are very helpful for maintenance team as well for customers, which save the cost and time of both. After that validation of requirement and design is added, this enhances the capability of this model as compared to previous model at the time of software development.

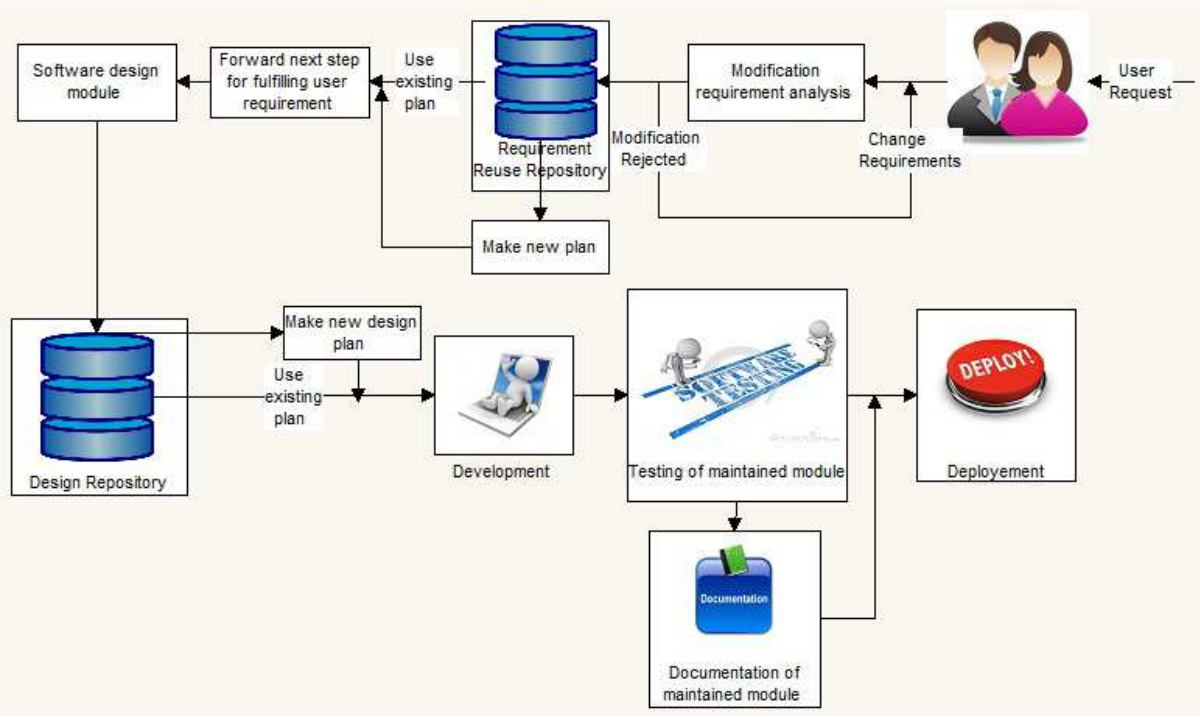

Fig. 1. Enhancement in Task-Oriented Software Maintenance Model by Using Requirement and Design Reuse Repository

In user maintenance request phase, a request is sent by user for performing maintenance activity on the already working software. A request can be made if system is not working according to expectations of users.

In modification requirement analysis stage a request from user already using the system is received. The modification can be adding new features, improvement in performance of existing modules, migration of system to another operational environment, modifying the existing working module, for correcting errors. The modification requests are then analyzed in detail for currently working system. These Changes may also be needed for the existing system to fulfill the working requirements of itself operational environment. After that the next step is "applicability of changed requirements", in this stage it is checked that the requirement which are demanded by customer are applicable or not. If the requirements are

applicable then we move towards next step. If the requirements are not applicable then we reject the requirement and move back towards previous step for changing requirements. Requirements are then also validated from customer for conformation. After checking the applicability of requirements, the requirements are then validated from user for conformation.

Component based software engineering plays a fundamental role for increasing the yield of an organization. This process requires a valuable and rich set of software components in the reuse repository. It is seen that, in most of software projects, once the requirements are gathered, the development process begins from scratch. This may lead our process to overtime and over budget. If the existing component or already gathered requirements are reused for developing or maintaining a new system rather than the developing the complete system from scratch, not only it saves the time but also it will produce a good quality product. The requirement repository is a place where bulk of already used requirements is stored. These bulks of requirements are then reused again for maintaining newly developed system. If the requirements for newly maintained system are not available in repository then requirements are gathered again and new plan is developed for maintaining this system.

Complete design of the existing system is needed to be understood completely by software maintainer in this phase of Program Comprehension. Source code of complete system is studied in this phase to perform the maintenance task. While surveying the work of software maintenance managers it is discovered that mismatched design documents of software are main cause of software maintenance problems. It is also notices that about 48% of problems in maintenance phase are due to poor participation of managers. If design documents are not available or reliable then it is needed to study source code alone. Commonly, there is no history of developed product design exists for performing maintenance task. Understanding of source code means, Reading code carefully, study the execution of program, and existing design documents.

The exact location in the system is indentified where the proposed modification has to be made, in this phase of Localization. It is very important to find out the exact area for modification in source code. This phase is also named as “spotting”. For this purpose, Design recovery or program slicing technique can be applied to the examined source code. It is important to put stress on the phase of program comprehension. This phase must be done completely and correctly before the proceeding of localization phase.

Impact Analysis is an important and difficult phase in software maintenance. Without proper attention to and mastery of the candidate system design, it would be hard to find out how the intended modification would cause side effects in the system, and where these would occur. After that, existing program and design document of already working software is studied for the sake of maintenance.

Component-based software development is an idea of building software system from already established software components. This technique opposes the building of software from scratch. In this phase, we study different components of software system on the basis of CBD (component software development) and CBSE (components based software engineering). Initially, some investment is required to start the software reuse process. In short, we can say that the development of a reuse process and reuse repository provides a basic of knowledge for software development that improves the quality of software module reuse cycle. This technique reduces the amount of software development. Ultimately, it decreases the risks of new projects because of repository knowledge.

The design repository is a place where bulk of already used design modules is stored. These bulks of design components are then reused again for maintaining newly developed system. If the design for newly maintained system is not available in repository then a new design plan is developed for maintaining this existing system. After that in Redesign phase, the existing system is then re-designed on the new request of customer. The design architecture is updated according to the proposed change in design phase. The development process begins after analysis phase and after accepting requirement and design phase. The development is implemented in iterations. The modified design object is then transformed into a working program.

Testing is the most crucial phase in the entire software maintenance process. In this phase ripple effects of modified components is tested for existing test cases, and the regression testing of maintained components is then performed. The maintenance testing approach is completed in following two steps:

- **IDENTIFICATION OF THE NEWLY ADDED, MODIFIED, OR DELETED MODULES.**

Second phase is Re-designing and re-running of the maintained test cases.

- **DOCUMENTATION OF MAINTAINED MODULE**

  In this phase, after testing the newly maintained module, we have to make documentation of this changed module. Because, documentation is very helpful to completely understand the working of this module and it is also helpful for re-maintaining this system again.

Deployment is the last phase of software development. This is the stage at which we put our developed software in working environment.

4 BENEFITS OF DEVELOPED MODEL

There can be many benefits of developed model for customers as well as for developers also. However, there are some important benefits of proposed model.

4.1 PROPER USER INVOLVEMENT

User involvement is treated as key feature in proposed model. User involvement and acceptance from users in various stages is considered as most important factor.

4.2 CUSTOMER SATISFACTION

Customer requirements and customer satisfaction are understood and taken at priority. However, hidden requirements are also elaborated in this model. This thing results in increasing customer satisfaction.

4.3 VERIFICATION AND VALIDATION OF REQUIREMENT AND DESIGN

This model provides validation and verification at requirement as well as at design phase. Requirement is a technique which is used in this model which helps customers as well as developers to validate and verify the requirements. After validation and verification of requirement from customers, design is also validated and verified from customers by showing them the prototype model of particular program. This thing is very beneficial for customer and developer also.

4.4 EFFICIENCY

This proposed model is so much clear in its phases and hence it is expected that it will provide efficient output. Because, efficiency plays a major role in the development as well as in maintenance stage of a software product.

4.5 TIME AND COST EFFECTIVE

Hence by using reusable requirement and design components a lot of time and cost in maintaining a software product is saved by reusing requirement and design components. A reusable requirement and design repository is prepared which is so much effective on saving cost and time of customer and developers in a positive way.

4.6 MANAGING PROJECTS IS BETTER

The projects with larger scale having lots of functionalities are sometimes difficult to manage. But by using this model large scale projects will be handled easily by dividing the work into small teams. The dedicated team members working under the supervision of a specialist leader should be able to perform quickly and efficiently.

4.7 LESS PRESSURE ON DEVELOPERS

By using reusable requirement and design modules the maintenance phase becomes easier. Because, by reusing already built modules much more effort decreases. Hence, lesser pressure is built on developers and also on team leader which are dedicated for maintaining software.

5 CONCLUSION AND RECOMMENDATIONS

The key objective of this research was to introduce an enhanced software maintenance process model. This research is expected to be the helpful for decision making so that user who is in software development field could take benefit from this model and research. For this purpose almost all the previous models have been studied and discussed with experts. This research concluded that:

The previous software maintenance models have some deficiencies like process was not so good according to the today's changed environment. Process was too complicated. Most of the customers don't have satisfactory results from previous maintenance models. After analyzing the customer and developers trend an Enhancement is software maintenance process model was proposed whose responses are satisfactory. As future work, this model can be applied to different case studies for coping with issues of usability and performance.

REFERENCES

[1] R. Kumar and N. Dhanda, "Maintainability Qualification Of Object Oriented Design: A Revisit", *International Journal of Advanced Research in Computer Science And Software Engineering*, vol.4, no.12, pp.461-466, 2014.

[2] Rizvi, S.W.A., & R.A Khan, "A Critical Review on Software Maintainability Models", *Conference on Cutting Edge Computer and Electronics Technologies*, Pantnagar, pp. 144 – 148, 2009.

[3] E.Z. Abdulrazzak and I. Ghani, "Secure Software Design Maintenance Using Enhanced Task-Oriented Security Maintenance (TOSIM) Model", *Science International (Lahore)*, vol. 25, no.3, pp. 235-244, 2013.

[4] Khan, M.K & M.A Rashid, "A task-oriented software maintenance model", *Malaysian Journal of Computer Science*, Vol.9, no.2, pp.36-42, 1996.

[5] Htoon, C.Y., & N.L Thein, "Model-based Testing Considering Cost, Reliability and Software Quality", *Proceeding of 6th Asia-Pacific Symposium on Information and Telecommunication Technologies*, Yangon: 160-164, 2005.

[6] A.Tahir and R. Ahmad, "An AOP-Based Approach for Collecting Software Maintainability Dynamic Metrics", *International Conference of Computer Research and Development*, Kuala Lumpur, vol. 2, pp. 168-172, 2010.

[7] R. Bayer and A.E Milewski, "Improving Software Maintenance Efficiency With Behavior-Based Cognitive Models", *Proceedings of IEEE International Conference on Systems, Man and Cybermetics*, Singapore, pp. 3353-3358, 2008.

[8] J. Buckley, "Requirements-Based Visualization Tools for Software Maintenance and Evolution", *IEEE Computer Society*, vol. 42, no. 4, pp. 106-108, 2009.

[9] M. Fei., T. Shu-rong and L. Bo, "Reuse-oriented Information Model of Product Design Process Module", *International Conference on Management and Service Science,* Wuhan, pp. 1-4, 2009.

[10] A. Tarvo, "Mining Software History to Improve Software Maintenance Quality: A Case Study", *IEEE Computer Society*, vol.26, no.1, pp. 34-40, 2009.

[11] V. Nguyen, "Improved Size and Effort Estimation Model for Software Maintenance", *International Conference on Software Maintenance,* Romania, pp.1-2, 2010.

[12] W. Chengjion, "The Reusable Software Component Development Based on Pattern-Oriented", *International Conference on Computational Intelligence and Communication Networks,* Mathura, pp. 552-555, 2012.

[13] B. Kumar, "A Survey of Key Factors Affecting Software Maintainability*", Proceeding of the IEEE International Conference of Computing Science (ICCS),* Phagwara, pp.261-266, 2012.

[14] P. Siddhi and V.K. Rajpoot, "A Cost Estimation of Maintenance Phase for Component Based Software", *IOSR journal of Computer Engineering,* vol.1, no.3, pp.1-8, 2012.

[15] M. Islam and V. Katiyar, "Development of A Software Maintenance Cost Estimation Model: 4TH GL Perspective*", International Journal of Technical Research and Applications*, Vol.2, no. 6, pp.65-68, 2014.

[16] V.B Singh and M. Sharma, "Prediction of The Complexity of Code Changes Based on Number of Open Bugs, New Feature and Feature Improvement", *International Symposium on Software Reliability Engineering Workshops*, Naples, pp. 478-483, 2014.

[17] J. Shi., Xu, L., B. Hao., & Z. Fan, "A Task Scenario oriented Test Requirement Modeling Method for Complex SOA-based System", *Annual IEEE International Systems Conference*, Vancouver, BC, pp.103-108, 2015.