International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

Developing Software Maintenance Workbench

Lokhande Rahul[1], Waychal Pradeep[2]

[1]Department of Computer Science and Information Technology, College Of Engineering Pune,
Wellesely Rd, Shivajinagar, Pune - 411 005, Maharashtra, India

Abstract: ***This document gives overview of Maintenance work. There is so much necessity of maintenance. It explains different types of maintenance. Different Causes of defects. How to handle these defects? And explains how to teach defect fixing? Main purpose is to reduce maintenance cost and making maintenance easy. Following this technique makes maintenance easy.***

Keywords: Software maintenance, Defect fixing.

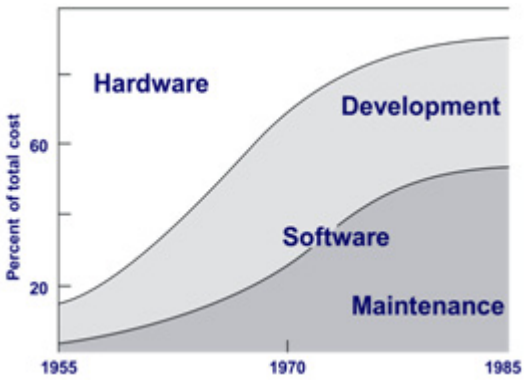
1. Introduction

In today's world use of computer, mobile, laptop, tablet is tremendously increasing.in previous days computer was only used for military and research purpose. Huge progress in computer and other electronics devices such as mobile phone, tablet, and laptop made these devices very cheaper [3]. And also low cost and high intelligence made these devices to be used in most of the industries and almost all sections. Due to high demand and verities in software there are certain bugs in software, which are necessary to be fixed. Some defects can be fixed easily but some are hard to detect as well as to fix.

Today many researches are going on finding the defects in given software. Students are taught to ways of finding defects in software but actual practical knowledge of student is low compared to theoretical one [5]. We are developing a system which adds defects in software and students will fix those defects using some previous knowledge. In this way student will learn how to find different defects. And practicing of finding defects will speed up their ability of finding defects.

2. Maintenance

Software maintenance: software maintenance is the process of modification of a software code after delivery of product in order to correct faults. In previous days about 85 % of total cost was spent on hardware. Near about 10 % of total cost spent on software and very few amount was spent on maintenance [8]. That means maintenance was given less important. It affects quality of software. Maintenance is very important part of software lifecycle. Later on software maintenance got high importance. To improve quality of software maintenance is become one of the most important units.



Maintenance can be classified into four classes:

2.1 Corrective maintenance

Corrective maintenance includes correcting existing errors [10]. If for some functionality software is giving wrong output then those errors should be fixed. It is practically not possible to test entire functionality of large software system. There it is obvious that system will have defects.

2.2 Adaptive maintenance

In this maintenance changes are related to software environment such as DBMS, OS [4]. In adaptive maintenance changes are related to OS and DBMS, if environmental requirement is not specified in advanced and client is not sure about in which environment the software is going to be used. Then this kind of defects occurs. For change of environment and OS, program has to be changed in most of the part. To avoid this ask client about exact use of software

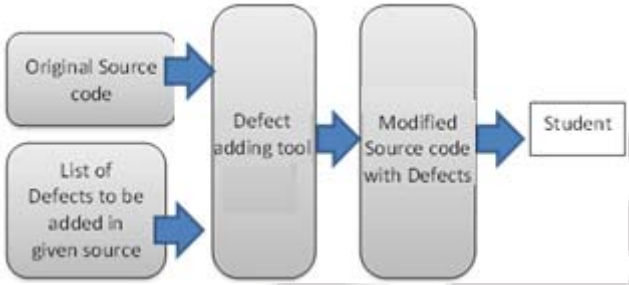
2.3 Perfective maintenance

Clients are never satisfied, they always want enhancement in existing code so that the product will become perfect in all aspects [5]. Perfective maintenance has to perform in order to make software perfect in all aspects. Sometime changes are minor, they do not effect on performance but still client want them to be fixed for better quality e.g. GUI related problems.

2.4 Preventive Maintenance

Some changes have no direct effect on the user, but these changes will help for future maintenance [1]. These changes will increase reliability for better future development. Technology changes day by day. So in order to walk with the technology we have to change some part of the code so that to avoid future maintenance.

3. System Design and Implementation



3.1 Original Source Code

The source code should be big enough so that finding and fixing the defects should not be easy task. And number of defects we can add may be considerably large.

3.2 Finding Defects

First task is to find defects in given code [2]. We studied different types of defects in given area. After that we found these types of defects in given code.

3.3 Adding Defects

As we have source code and list of defects we can add defects easily. Scan list of defect line by line [11]. And for each line open file and add defects into the file. To remove defects from file we need to study different types of defects? Why they occur? What is its effect on the performance and accuracy of the code? How to remove these defects? Which tools are useful?

3.4 Verifying results:

Now check the original defect file with the defect file submitted by student [8]. How many defects he has found? How many defects he missed? Is there any new defect which is already exists in original source code. Now practicing this way will increase student's ability to find defects.

4. Defects Covered

4.1 Un-initialized variable

In C language if variable is not initialized the compiler will initialize it to any random value. Thus program will get compiled successfully i.e. it will not throw any error. But it will show incorrect output.

4.2 Safe Synchronization

If any resource is locked before its use then it should be unlocked after its use otherwise it may lead to deadlock so it is necessary that each LOCK should be always followed UNLOCK.. If one LOCK is not followed by UNLOCK then resource is not freed so other process let's say Process A will not be able to use this resource though resource is not busy. So the A process will wait indefinitely for that resource. This leads deadlock

4.3 Memory Leak

Memory leak is an important defect in software mainly in operating system [9]. It occurs if memory is allocated but not released after its job is over. If memory is not released after its job is over then a small amount of memory is still busy and it cannot be allocated by system again. In this way each time some memory is wasted. At a particular instant, there is no free memory available to allocate. In this case system will hang.

4.4 Null Pointer Dereferencing

If memory is allocated for some variable and assigned to any pointer then before assigning value to variable, it should be checked that value of pointer is not null [6]. If in some case malloc fails then it will return NULL and we cannot assign any value to memory address null. Many programmers allocate memory and directly assign value to variable without checking whether memory is allocated or not. Many of them assume that each time memory will get allocated but it is not the case always.

5 Conclusion and Future Work

It is observed that finding defect in given code is very challenging task. It takes too much time for finding defect in code for new person, though he has basic knowledge of defect. It happens because of lack of practical knowledge and training. After using this system students were initially taking too much time for each defect. But after more practicing, the time required to find defect is reduced considerably. So it increased students' ability to find defect and increased their logical thinking. In future work more types of defect can be added, so that it will cover almost all types of defects, finding defect is an time consuming task it may be done automatically. Also checking whether defects are properly fixed or not, may be checked automatically.

References

[1] Code decay - Stephen G. Eick, Todd L. Graves, Alan F. Karr, J.s. Marron, and Audris Mockus, "Does code decay? Assessing the evidence from change management data", IEEE Transactions on Software Engineering, 27(1), pages 1-12, 2001.
[2] Suresh Kothari, Ahmed Tamrawi, Kang Gui, "Event Flow Graphs to Verify Absence of Vulnerabilities and Malicious Behaviors", IEEE transactions on software engineering, manuscript id

[3] Timo Koponen "Evaluation of Maintenance Processes in Open Source Software Projects through Defect and Version Management Systems", ISBN 978-951-781-988-6
[4] S. Neginhal and S. Kothari, "Event views and graph reductions for understanding system level c code," in ICSM '06: Proc. of the 22nd IEEE International Conference on Software Maintenance, 2006, pp. 279–288.
[5] K. Gui and S. Kothari, "A 2-phase method for validation of matching pair property with case studies of operating systems," in IEEE 21st
[6] Sigmund Cherem, Lonnie Princehouse and Radu Rugina, "Practical Memory Leak Detection using Guarded Value-Flow Analysis" ACM New York, NY, USA ©2007 Pages 480-491
[7] Bjarne Steensgaard, "Points-to analysis in almost linear time. In Proceedings of the ACM Symposium on the Principles of Programming Languages", St. Petersburg Beach, FL, January 1996.
[8] Cherem, S., Princehouse, L., and Rugina, R. (2007a), "Practical memory leak detection using guarded value-flow analysis". In Proceedings of the 2007 ACM SIGPLAN conference on Programming language design and implementation, PLDI '07, pages 480{491, New York,NY, USA. ACM.
[9] Yulei Sui Ding Ye Jingling Xue, "Static Memory Leak Detection Using Full-Sparse Value-Flow Analysis", ISSTA '12, July 15-20, 2012, Minneapolis, MN, USA.
[10] Christopher J. Rossbach, Owen S. Hofmann, Donald E. Porter, Hany E. Ramadan, "TxLinux: Using and Managing Hardware Transactional Memory in an Operating System" ACM New York, NY, USA ©2007, Pages 87-102
[11] David Hovemeyer, Jaime Spacco, and William Pugh, "Evaluating and T uning a Static Analysis to Find Null Pointer Bugs", ACM New York, NY, USA ©2005, Pages 13 – 19

Author profile

Mr Rahul K. Lokhande received a B.Tech from Modern College of Engineering Pune. He is currentely M.Tech student at College of Engineering,Pune,India.

Dr Pradeep K. Waychal received a B.E. from College of Engineering, Pune, M. Tech from IIT Delhi and Ph.D from IIT Bombay. He is currently working as professor at college of Engineering, Pune, India.