Developing Software Maintenance Workbench

Dissertation

Submitted in partial fulfillment of the requirement for the degree of Master of Technology in Computer Engineering

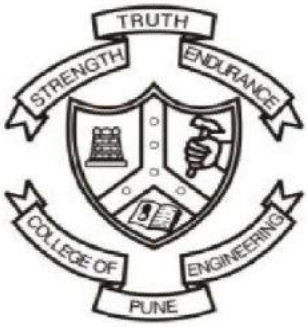
By

Rahul K. Lokahnde

MIS No: 121222002

Under the guidance of

Dr. PradeepWaychal



Department of Computer Engineering and Information Technology

College of Engineering, Pune

Pune – 411005

June, 2014

DEPARTMENT OF COMPUTER ENGINEERING AND

INFORMATION TECHNOLOGY,

COLLEGE OF ENGINEERING, PUNE

CERTIFICATE

This is to certify that the dissertation titled

Developing Software Maintenance Workbench

has been successfully completed

By

Rahul K. Lokhande

MIS No: 121222002

and is approved for the partial fulfillment of the requirements for the degree of

Master of Technology, Computer Engineering

Dr. PradeepWaychal
Project Guide,
Department of Computer Engineering
and Information Technology,
College of Engineering, Pune,
Shivaji Nagar, Pune-411005.

Dr. J. V. Aghav
Head,
Department of Computer Engineering
and Information Technology,
College of Engineering, Pune,
Shivaji Nagar, Pune-411005.

June 2014

Acknowledgments

I express my sincere gratitude towards my guide Professor **Dr. PradeepWaychal** for his constant help, encouragement and inspiration throughout the project work I would also like to thanks **Prof. Suresh Kothari**, Iowa State University, and **Jon Mathews,** Senior Software Engineer, EnsoftCorp.for providing directions to make progress in this work. Without their valuable guidance, this work would never have been a successful one.

I would also like tothank HOD **Dr. J. V. Aghav** as well asall the faculty members and staff of Computer and IT department for providing us ample facility and flexibility. Last, but not the least, I would like to thank my classmates for their valuable suggestions and helpful discussions. I am thankful to them for their unconditional support and help throughout the semester.

Lokhande Rahul K.

College of Engineering, Pune

Abstract

Software maintenance is modification of software code after submitting product to client to improve performance, speed of software. Changing the code means fixing the defects. Defect in software causes misleading behavior of product which might not be detected in testing phase.

In previous days, software maintenance has not given so much importance. But in now a days software maintenance became an essential part of software maintenance. As quality of software got huge importance now a days about 75% amount of total cost is spent on maintenance. Researchers say that about 60% of maintenance e ort is taken in understanding the software. In maintenance considerable time is spent on reading programs in order to implement changes. In maintenance it requires to identify system components and their relationship with each other. For good maintenance call graphs and control flow graphs are very useful.

To learn fixing different types of defects, first we need to understand different defects, how they can be added? What are their effects on the code? How it leads misbehavior? We are taking a sample code mute that code so that it contains some defects. We are adding different types of defects into that code, such as memory leak, safe synchronization. Now this code is given to new student or person who is unaware of detecting defects. And teach him how to find defects. After this student will find out defects and fix it.

To find different defects in given code some tools are very useful. C atlas is one of them. It shows call graph of mentioned function. So that understanding of flow of code gets simple. It is useful if code is too large and scanning code line by line is not easy and tracing calls of functions is critical task. In this way this tool is useful for teaching students how to find defects. Our main aim is to add di different types of defect in given software.

Table of Contents

Table of Figures

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. And also low cost and high intelligence made these devices to be used in most of the industries and almost all sections.

In financial section such as banking, Stock Exchange, online shopping, online recharge, online courses use of computer is increasing. In banks main use is for money transfer using mobile banking, internet banking, SMS banking, core banking. These facilities make huge contribution in increasing banks business and satisfy customer's requirement. The actual transfer of money happens later. But till then money is deducted from your main account balance. To provide this facility we need good quality of software which will handle all problems related to failure such as low balance, power failure, security. In stock exchanges, previously people were not be able to do online trading but now a days many people are doing online trading only because of high growth in IT industry. This increases load over the systems. Systems should be able this much amount of load.

Use of computer is also increased in medical field. Many of the medical tests are done using computer within few hours. In this case accuracy and speed are important factors. Any failure in this case causes high risk with patient treatment.

In research and defense computer plays an important role. If missile targeted at particular destination is not accurate, it has some defects then it will miss the target and will hit wrong object. This causes high disaster. If flight system hangs between its journeys, then it may cause catastrophic disaster.

1.1. Motivation

With the huge increase in IT industry, use of software is also increased. In each area software is becoming important part. Use of software is increasing day by day in automobile, research, pharmacy, defense and many more. Each industry and firm are using software for automation purpose, database handling purpose and data management purpose.Company's most work depends on software.

Each industry spends so much money on software. Obliviously each expects fewer defects in provided software. But it doesn't happen. Each software is having defects which were not detected in testing phase. So there is need of software maintenance. Maintenance should be provided so that the defects detected later should be solved and software will perform well for long. Lots of defects decrease quality of software.

Finding defect in given software is very difficult, challenging and time consuming task. Most of bugs are removed in testing phase. Some bugs cannot be detectedbe detected at testing level. These bugs are called as defects. Generally developer and maintenance provider are different people. So the maintenance people have no previous knowledge about given software. They need to study software from start to end. Most time is passed for understanding the software. Then finding the defect and then fixing it.

In this way software maintenance is really need of software lifecycle. Company should give more focus on software maintenance.

2.Problem Definition

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. 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.

3.Terms and concepts

3.1.1. Error:

If the actual result is different than expected result then it is called as error in software. Errors present in program because of confusion of the concept, poor knowledge of the programming language.

3.1.2. Failure:

The inability of a system or component to perform its required functions within specified performance requirements.

3.1.3. Bug**:**

A fault in a program which causes the program to accomplish in an unintended or unexpected manner.

3.1.4. Fault:

An incorrect step, procedure, or data definition in a computer program which causes the program to perform in an unintended or unanticipated manner

3.1.5. Defect:

Commonly refers to several troubles with the software products, with its external behavior or with its internal features.

4.Literature survey

4.1. Maintenance

Software maintenance: software maintenance is the process of modification of a software code afterdelivery of product in order to correct faults

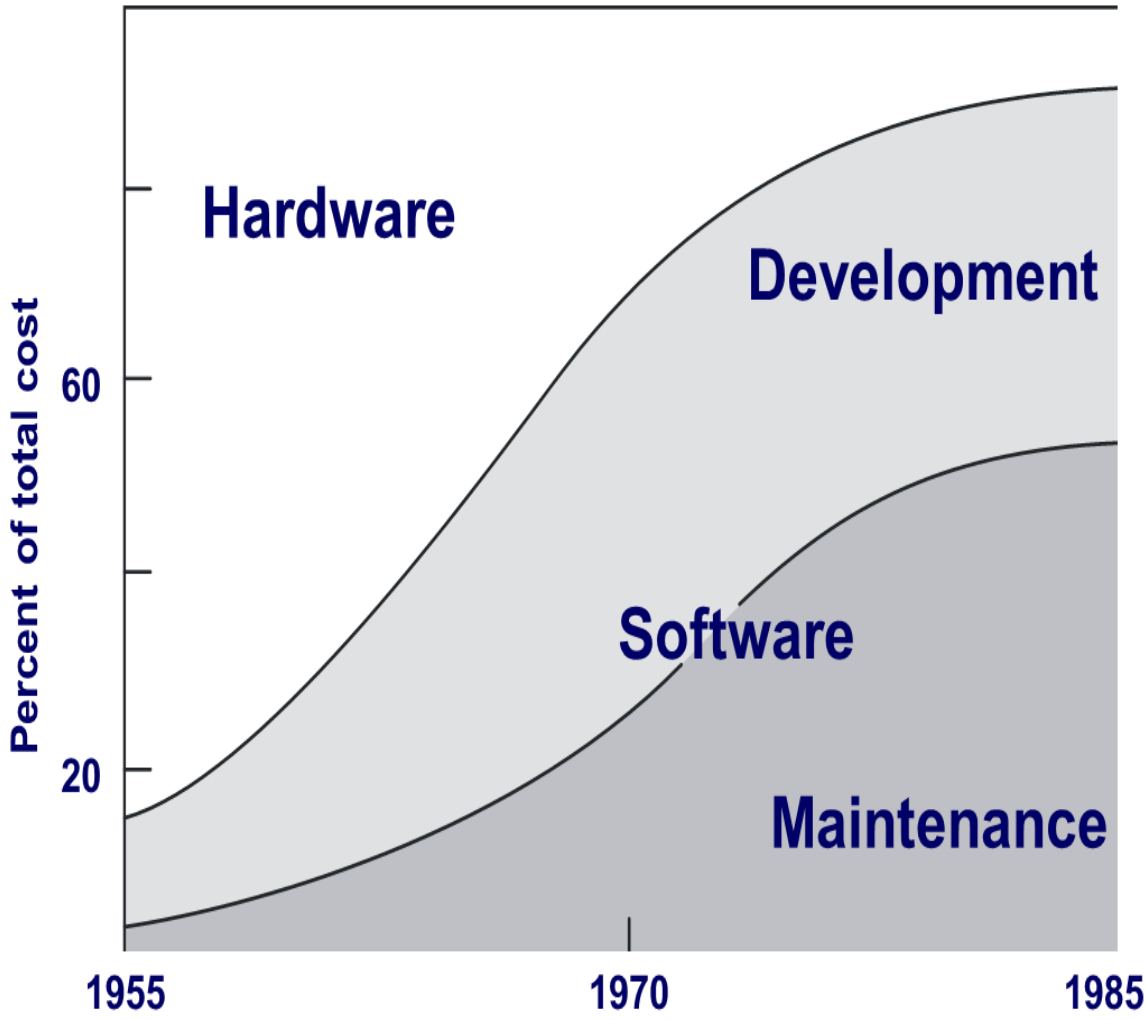


Figure 1- Cost estimation

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. 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 Maintenance can be classified into four classes:

4.1.1. Corrective maintenance:

Corrective maintenance includes correcting existing errors. If for some functionality software is giving wrong output then those errors should be fixed. It is practically not possible to test each and every functionality of large software system. There it is obvious that system will have defects. Corrective maintenance mainly concerned with correcting errors exists in the software. These errors occur only because of mistake done by programmer. This mistake occurs because of either misunderstanding of problem by programmer or lack of programming knowledge of programmer. The responsible person for this kind of maintenance is developer. So it is necessary that programmer should have strong knowledge of programming language and also he/she is clear about idea of software.

4.1.2. Adaptive maintenance:

In this maintenance changes are related to software environment such as DBMS, OS. 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. Explain him what will happen is OS is changed and Database. What are the advantages and disadvantages of each kind of OS and Database? If client totally agrees totally then only start development.

4.1.3. Perfective maintenance:

Clients are never satisfied, they always want enhancement in existing code so that the product will become perfect in all aspects. 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.

4.1.4. Preventive maintenance:

Some changes have no direct effect on the user, but these changes will help for future maintenance. 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. Suppose any software works on .jpg, .png formats of the file but later on some new format comes in market. Then to make software compatible with this format changes have to be made in order to avoid future changes.

Many researcher works on finding defects in given code and used different techniques to find different defects.

Characterization of different types of design defects and present symptoms is not necessary to detect defect. To generate detection rules examples of design defects is used. These detection rules are used in correction is used. The main method used is combination of refactoring operations. Fitness function is calculated after applying refactoring and defect count using detection rule.

5.System Design and Implementation

5.1. Different types of defects

5.1.1. 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. So we need to check each variable which is used in RHS or printf function should be initialized before its use.

e.g

```
main()
        {
        int index, sum, lastCount=10;
                for( index = 0 ; index <lastCount ; index++)
                        {
                        sum += i;
                        }
        }
```

As shown in above code sum is not initialized, still program will execute and will give wrong answer. To fix these kind of defect first store each variable in table with count as zero, now scan code line by line if variable occurs in right side of an expression and count of variable is zero then report there is defect. If variable occurs at left side of expression then set its count to 1.

5.1.2. 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, in same way resources held by process A will also not be released. So at the end system will enter in deadlock condition. To check this kind of defect we have to check each execution path of which starts from acquireLock() function now each execution path must contains releaseLock() function else there is no safe synchronization. It is very difficult to detect that the problem occurred is because of safe synchronization defect.

5.1.3. Memory Leak:

Memory leak is an important defect in software mainly in operating system. 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. If it is desktop then we can reboot it but our ongoing work will be erased. But what happens if it website on server .if the system is Flight control system and it hangs in middle on journey. In this way memory leak is very important defect and need to be fixed.

Detecting this problem in software testing does not ensure that the defect will be detected. Testing goes through one path at a time. As there are tremendously large number of execution paths. It is impossible to go through each path.

Also this defect may not show its presence for long time because it occurs if any function leaking memory will be called again and again. This happens rarely.

5.1.4. Array index out of bound:

In C array bound is responsibility of programmer. If by mistake array index reaches out of its bound then it must show error message if not then it is a defect. So for each array it should be checked that index should not exceed array size. If index of array reaches out of bound then this case should be handled by programmer correctly. Else this can lead to be serious problem; if operation is performed on multidimensional array then it increases possibility of occurring this kind of defects.

This kind of defect leads to loss of data. It is required that this defect should be fixed in early stages.

5.1.5. 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. 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. Sometime malloc will fail i.e. if there is no free memory for allocation. That's why there is need to check the pointer returned by malloc before assigning value to it.

5.2. Xv6 Exploration

Xv6 is a simple Unix-like Operating System developed in 2006 at MIT. It is simple operating system mainly designed for teaching purpose. Its code is about 8000 lines and it includes all basic functionalities of an OS. Source code is freely available on MIT's website. Source code is written in C language. Code should be complied using GNU compiler. It needs QEMU emulator for execution of code.

There are some defects those occurs in operating system and need to fixed such as memory leak, synchronization, and this is very challenging defect that's why we have chosen this OS, so that student will familiar with OS defects. Execution of this OS occurs on linux machine with QEMU emulator.

5.3. Tool Used

5.3.1. C – Atlas

C atlas is used for analyzing any C language code. It provides flow graph between mentioned functions. So if there is large number of functions then and for particular defect there is no need to analyzeall functions thus this tool reduces the unnecessary overhead of going through all code for particular defect. Also it gives direct linking between linked function then it is easy to go from one function to other.

5.4. Program Flow

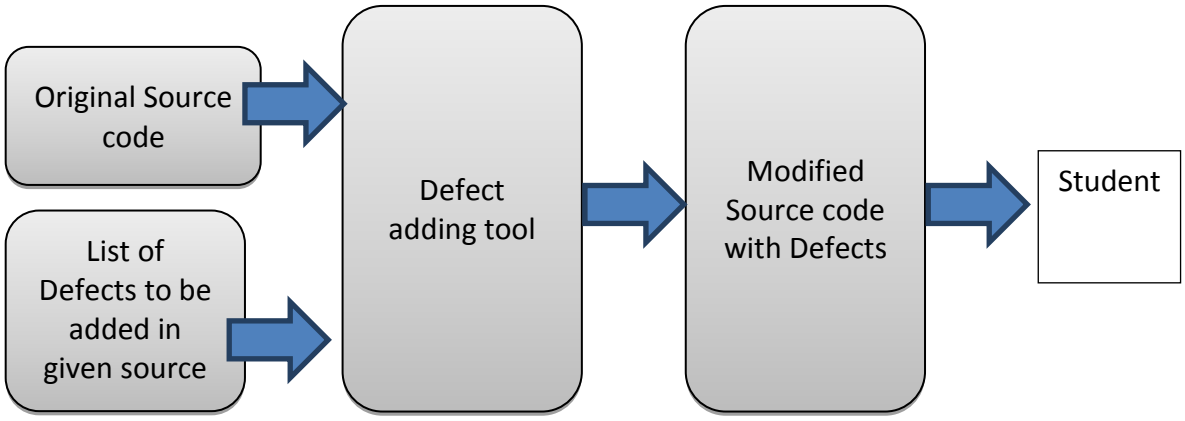


Figure 2Flow Daigram

5.4.1. 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.

5.4.2. Finding defects:

First task is to find defects in given code. We studied different types of defects in given area. After that we found these types of defects in given code. And stored into .xlsx file and the file is converted into .csv file with field separator as tab. File contains four columns first is filename, second is line number from that file, third is line before modification fourth one is line after modification.

5.4.3. Adding Defects:

As we have source code and list of defects we can add defects easily. Scan list of defect line by line. And for each line open file with name same as mentioned in column 1 (file name). Now go to line number mentioned by column 2 (line number). And replace that line by line mentioned in column 4 (modified line).

We have two codes original source code as well as modified source code. Now keep original source code and give modified source code to students. First of all teach them 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? All these things should be taught to student. After this students will find defects in code and will modify the code as per need or just mention line numbers and what is the issue with the code, what type of defect it is? What changes should be made?

5.4.4. Verifying results:

Now check the original defect file with the defect file submitted by student. 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.

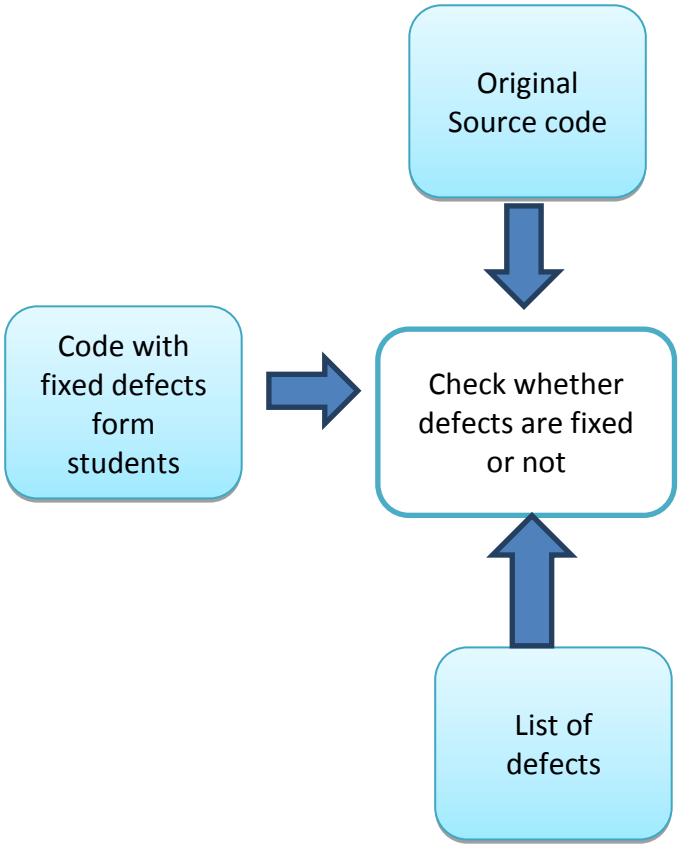
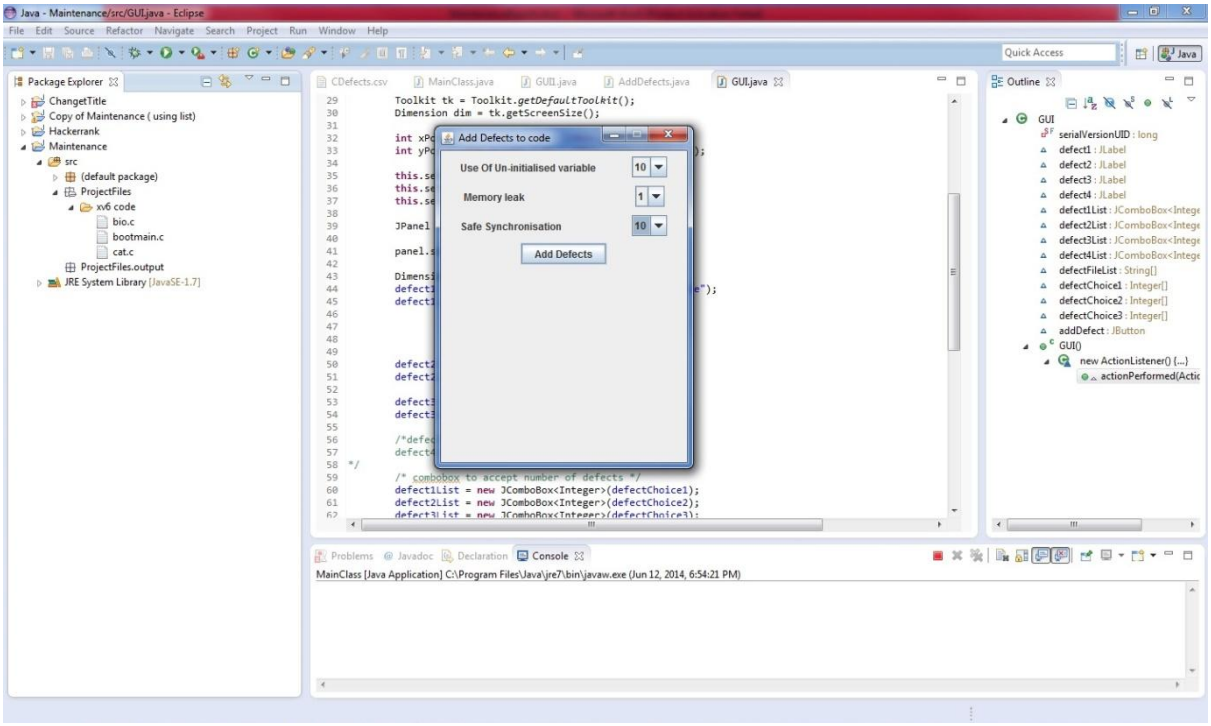


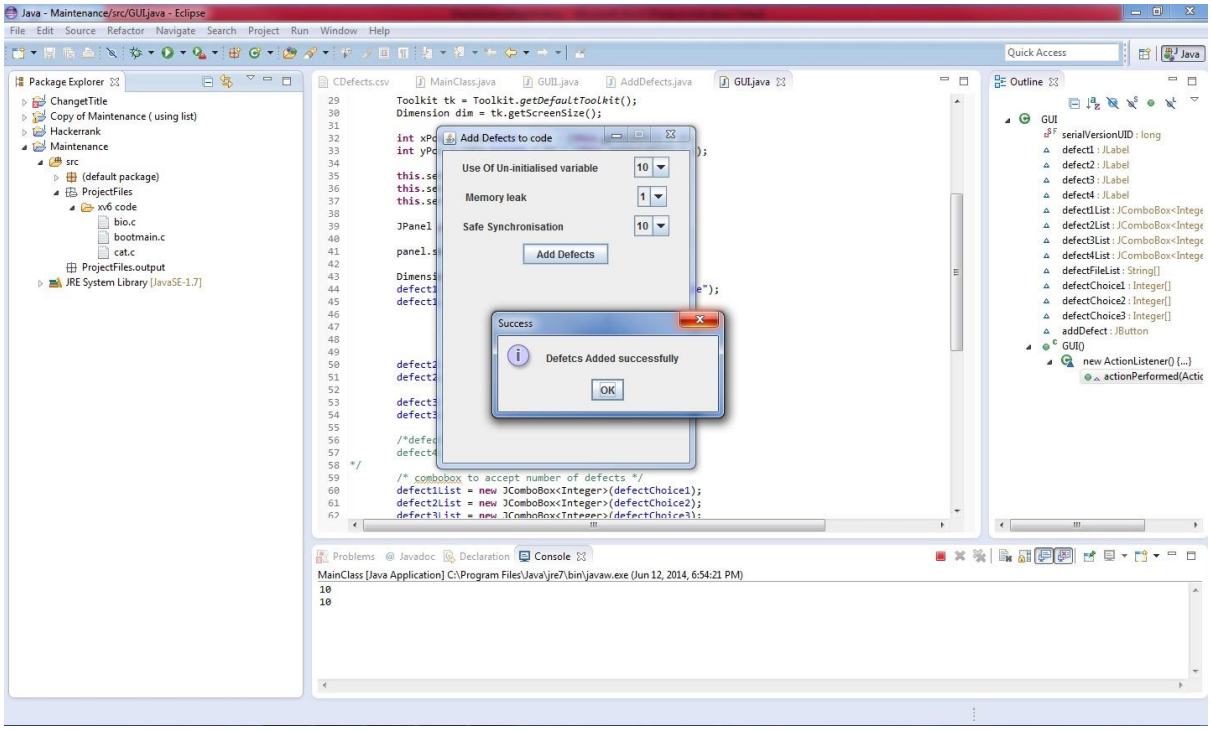
Figure 3-Varifying Defects

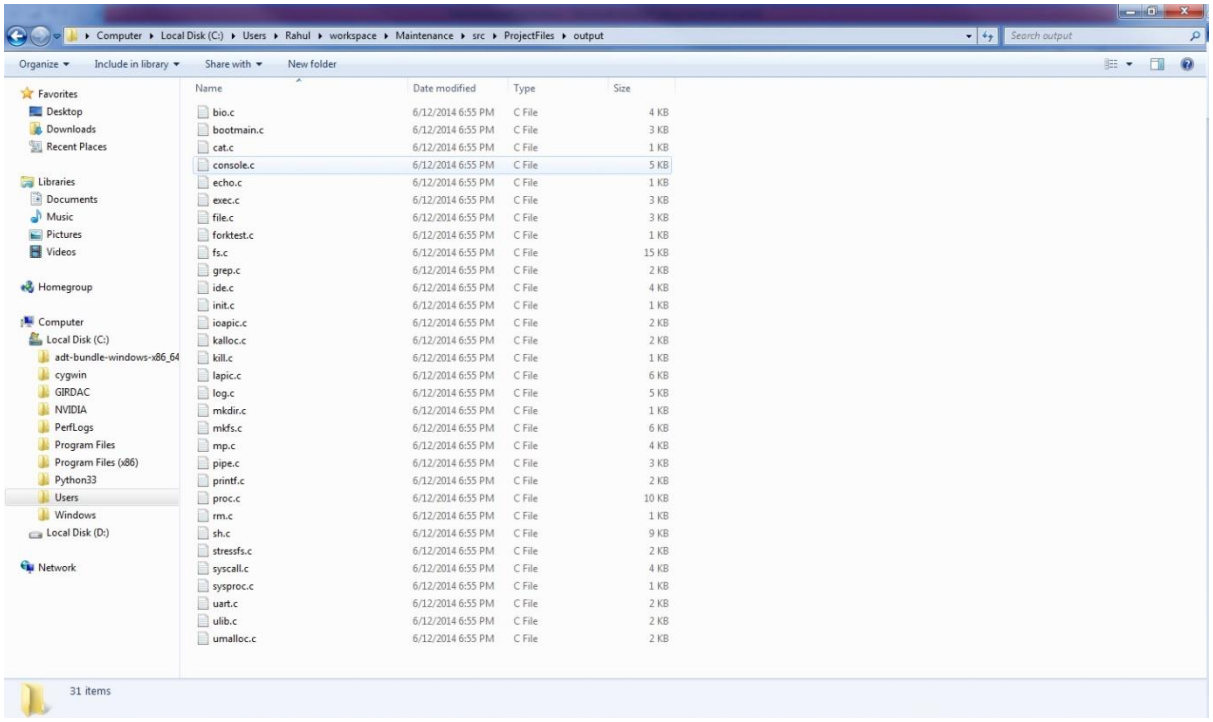
Format to add defect

File name	Line no	Before	After
bio.c	66	acquire(&bcache.lock);	
bio.c	125	acquire(&bcache.lock);	
console.c	62	acquire(&cons.lock);	
console.c	192	acquire(&input.lock);	
console.c	235	acquire(&input.lock);	
console.c	271	acquire(&cons.lock);	
file.c	30	acquire(&ftable.lock);	
file.c	46	acquire(&ftable.lock);	
file.c	60	acquire(&ftable.lock);	
fs.c	227	acquire(&icache.lock);	

As shown in fig defect list contains file name, line number, before and after columns. After executing code first file bio.c opens and at line number 66 acquire (&bcache.lock); is replaced by blank i.e. we are removing acquire lock function now bcache is not locked so any other process can also access bcache which is not expected in this way defect is added into code


Java - Maintenance/src/GUI.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Package Explorer
ChangetTitle
Copy of Maintenance (using list)
Hackerrank
Maintenance
src
(default package)
ProjectFiles
xv6 code
bio.c
bootmain.c
cat.c
ProjectFiles.output
JRE System Library [JavaSE-1.7]
CDefects.csv
MainClass.java
GUI1.java
AddDefects.java
GUI.java
Toolkit tk = Toolkit.getDefaultToolkit();
Dimension dim = tk.getScreenSize();
Add Defects to code
Use Of Un-initialised variable
10
Memory leak
1
Safe Synchronisation
10
Add Defects
/* combobox to accept number of defects */
defect1List = new JComboBox<Integer>(defectChoice1);
defect2List = new JComboBox<Integer>(defectChoice2);
Outline
serialVersionUID : long
defect1 : JLabel
defect2 : JLabel
defect3 : JLabel
defect4 : JLabel
defect1List : JComboBox<Intege
defect2List : JComboBox<Intege
defect3List : JComboBox<Intege
defect4List : JComboBox<Intege
defectFileList : String[]
defectChoice1 : Integer[]
defectChoice2 : Integer[]
defectChoice3 : Integer[]
addDefect : JButton
GUI()
new ActionListener() {...}
actionPerformed(Actic
Problems
Javadoc
Declaration
Console
MainClass [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (Jun 12, 2014, 6:54:21 PM)


Java - Maintenance/src/GUI.java - Eclipse
Add Defects to code
Use Of Un-initialised variable
Memory leak
Safe Synchronisation
Add Defects
Success
Defetcs Added successfully
OK
10
10


Computer ▸ Local Disk (C:) ▸ Users ▸ Rahul ▸ workspace ▸ Maintenance ▸ src ▸ ProjectFiles ▸ output
Search output
Organize
Include in library
Share with
New folder
Favorites
Desktop
Downloads
Recent Places
Libraries
Documents
Music
Pictures
Videos
Homegroup
Computer
Local Disk (C:)
adt-bundle-windows-x86_64
cygwin
GIRDAC
NVIDIA
PerfLogs
Program Files
Program Files (x86)
Python33
Users
Windows
Local Disk (D:)
Network
Name
Date modified
Type
Size
bio.c 6/12/2014 6:55 PM C File 4 KB
bootmain.c 6/12/2014 6:55 PM C File 3 KB
cat.c 6/12/2014 6:55 PM C File 1 KB
console.c 6/12/2014 6:55 PM C File 5 KB
echo.c 6/12/2014 6:55 PM C File 1 KB
exec.c 6/12/2014 6:55 PM C File 3 KB
file.c 6/12/2014 6:55 PM C File 3 KB
forktest.c 6/12/2014 6:55 PM C File 1 KB
fs.c 6/12/2014 6:55 PM C File 15 KB
grep.c 6/12/2014 6:55 PM C File 2 KB
ide.c 6/12/2014 6:55 PM C File 4 KB
init.c 6/12/2014 6:55 PM C File 1 KB
ioapic.c 6/12/2014 6:55 PM C File 2 KB
kalloc.c 6/12/2014 6:55 PM C File 2 KB
kill.c 6/12/2014 6:55 PM C File 1 KB
lapic.c 6/12/2014 6:55 PM C File 6 KB
log.c 6/12/2014 6:55 PM C File 5 KB
mkdir.c 6/12/2014 6:55 PM C File 1 KB
mkfs.c 6/12/2014 6:55 PM C File 6 KB
mp.c 6/12/2014 6:55 PM C File 4 KB
pipe.c 6/12/2014 6:55 PM C File 3 KB
printf.c 6/12/2014 6:55 PM C File 2 KB
proc.c 6/12/2014 6:55 PM C File 10 KB
rm.c 6/12/2014 6:55 PM C File 1 KB
sh.c 6/12/2014 6:55 PM C File 9 KB
stressfs.c 6/12/2014 6:55 PM C File 2 KB
syscall.c 6/12/2014 6:55 PM C File 4 KB
sysproc.c 6/12/2014 6:55 PM C File 1 KB
uart.c 6/12/2014 6:55 PM C File 2 KB
ulib.c 6/12/2014 6:55 PM C File 2 KB
umalloc.c 6/12/2014 6:55 PM C File 2 KB
31 items

6.Conclusion& 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.

7.References:

[1] Code decay - Stephen G. Eick, Todd L. Graves, Alan F. Karr, J.s. Marron, and AudrisMockus, “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] TimoKoponen “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 RaduRugina, “Practical Memory Leak Detection using Guarded Value-Flow Analysis”ACMNew York, NY, USA©2007 Pages 480-491

[7] BjarneSteensgaard,“Points-to analysis in almost linear time.In Proceedings of the ACM Symposium on the Principles ofProgramming 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 conferenceon 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 DetectionUsing 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 TransactionalMemory in an Operating System" ACMNew 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", ACMNew York, NY, USA©2005, Pages 13 – 19

[12] David Hovemeyer, William Pugh, "Finding More Null Pointer Bugs, But Not Too many", workshop on Program analysis for software tools and engineering, ACMNew York, NY, USA©2007 Pages 9 - 14

[13] Cesar Couto, Pedro Pires, Marco Tulio Valente, Roberto S. Bigonha,NicolasAnquetil "Predicting software defects with causality tests", Science Direct The Journal of Systems and Software 93 (2014) 24–41

[14] Krishnan, M.S, Kellner, M.I.Software Engineering, IEEE Transactions on (Volume:25 , Issue: 6) 800 – 815

[15] Aura Yanavi, "Methods and systems for predicting software defects in an upcoming software release", patent US20050071807 A1

[16] Steven Sit, Daniel DeKimpe, TamuyenPhung, "Method and apparatus for processing information on software defects during computer software development" US7401321 B2

[17] MarcoDAmbros, Alberto Bacchelli, Michele Lanza ,” On the Impact of Design Flaws on Software Defects”, 2010 10th International Conference on Quality Software, IEEE