

EMPIRICAL STUDY OF GUI TO REUSE UNUSABLE TEST CASES

ALBERT MAYAN. J¹, LAKSHMI PRIYA. K², YOVAN FELIX. A³

^{1,3}Associate Professor, Department of Computer Science and Engineering, Faculty of Computing, Sathyabama University
²P.G Student, Faculty of Computing, Sathyabama University
E-mail: ¹albertmayan@gmail.com, ²priya.k449@gmail.com, ³yovanfelix@gmail.com

ABSTRACT

Now-a-days it is more important to upgrade web applications which are provided by companies due to changes in user requirements. In developing web applications the graphical user interface (GUI) plays vital role. Due to frequent changes in application it varies from old version and modified version. Finally, it makes some of the test cases as unfeasible in modified version. We mainly, focus in areas where modified version is affecting the old test suite to make some of the test cases as unfeasible. To rectify this problem, we used to make changes in input constraint values. Here, we consider regression testing to make testing more flexible whenever changes occurred. Testing can be done manually and also by using tools. By this technique we can make maximum number of unfeasible test cases to feasible in modified versions. By reusing the unusable test cases it requires less time to make modified version as more efficient and also reduces cost of regression testing.

Keywords: *Regression Testing, Feasible And Unfeasible Test Cases, Constraint Values, Web Applications.*

1. INTRODUCTION

To develop any software for desktop application the common approach is graphical user interface (GUI). On web applications wide numbers of companies are relayed due to more interactive to the users. Requirement of users changes frequently so it makes differences between versions. When varies occur it also make changes in source code. Due to this when testing occurs it forms some unfeasible test cases which cannot give expected output. In this regression testing for web applications developed a technique as making changes in input values. When the modification occurs it cannot reach the expected output. So, to make them useful in modified version we make changes in the constrain values. When the implementation has done, work of the testing will become complicated because of changes in source code testing has to be done frequently. So, we use regression testing which can repair and make unfeasible test cases as feasible test cases. Here, we mainly concentrate on parts where it affected due to modification. Then it can form test cases and verifies whether it can give actual output or not. This technique is used to reduce cost of

regression testing by making to reuse maximum number of test cases.

2. PROBLEM FORMULATION

It demonstrates a technique for test cases where code was affected and result faults in test cases. The complete code is taken in PHP form. Their main goal is to concentrate on affected part and they select such paths by dividing whole program into sub parts. Finally, they will test these test cases with different input values [1].

They proposed steps to assure modifications done according to user requirements or not. It derived LQN models for software's by designing UML diagrams. It is a step by step approach for converting software models as specifying UML diagrams. Finally, this is more useful for our approach to make defects to be corrected [2, 3].

It demonstrates a programming language works on server side to create web application dynamically with PHP. At the runtime many of errors are detected and these errors are declared as dangling references. It mainly, checks the paths of execution done in

programs where occurs at constrain paths by having matches in constrains. [4].

It demonstrates a framework for testing in PHP by focusing on language called PRASPEL. They generate automatically data for test cases which satisfy the conditions. For this purpose they made implementation for solving constrains with PHP which can make solutions to diverse ensuring test cases randomly. [5].

They proposed a tool to support the selections in test cases which mostly occurred in program designs by using regression testing to detect the defects. Here, they required unified modeling languages (UML) to select test paths in programs. By having tools called prototype changes in design are feasible [6].

It demonstrates a technique where the input value varies from old version and new modified versions. It is used to identify constrain values which are reusable in modified version where reusable test cases are taken from previous version. It can make less effort of regression testing [7].

They proposed a tool called CSCW for supporting innovation of natura campus open. The collaboration of systems is used for the test cases for applications because new products contain advanced customer requirements. This approach is more useful for research purposes for further enhancements [8].

It demonstrates a technique for model based generation by using UML diagrams. More effort of humans is need for representation of action sequences for abstract models. Here, active diagrams are implemented to apply on test cases obtained in modified application[9].

It demonstrates a technique for common constrains to make the length of keyword to be shortened by making their distances to skip by using pre filters. For this problem they made a solution called skipping distance for these keywords should be longer[10].

It proposed PHP scripts for protecting the issues like unwanted, copying, and modifying and so on. It also has a problem for not having enhancing version. For overcome these problems they developed a novel technique which can provide high security by cryptology [11].

It proposed a model called architectures driven model which in one of the technique to UML models for testing. It has high popularity because here the faults are detected and corrected. For this solution regression is best for making designs perfect and also to set rules [12].

It demonstrates a methodology to update and fix faults in test cases and also collect information from area covered. Finally, the result shows the range of hit range of cache. This is more effective and accurate deals with simple methodology for testing the system [13].

The technique is proposed for improvement in fault detection rate. They introduce prioritization technique which is powerful when modified versions are targeted. They also experimented between granularities where trade off exists between them [14].

It demonstrates a technique for code programming to reduce difficulties called as interline power flow controller. This model mainly deals with less power injections and Jacobean matrix for making series to be converted. This advanced version is used to point faults in systems installed [15].

They proposed a technique for programming language applications where they have dynamic behavior it analyses this on selected area. They proposed a system where the changes in software contains variation in versions so this is used to combine the two techniques which are code based and model based [16].

It demonstrates for software validation to select the test cases by using regression testing on the components which are updated on applications. They introduced prototype for parts where impacts occurred due to modifications. This strategy is used for maintainence of application [17].

3. METHODOLOGY

In our approach we deal with a strategy to reuse old test cases by giving different inputs and make use in modified versions. Existing system shows if any modification has been done in current application it may not satisfy all the requirements needed by user. To overcome this problem our work shows that instead of deleting to reuse them in the new version. Actually,

complete code cannot be affected only some of parts are only affected. So, in affected part the designed test cases can become unfeasible instead of avoiding them we reuse in modified code. To show the technique we designed a webpage where source code is written in PHP and testing is done and test cases are designed. For these test cases we check whether the conditions required are satisfied or not. The test cases which reach conditions can become feasible and others cannot. So, instead of avoiding we use them in modified code. This can be done as, by having same constrains but we make values which can be given are changed.

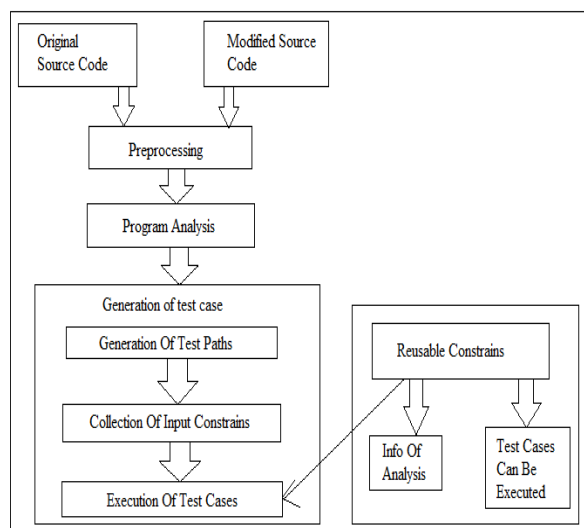


Fig 1: System Overview

The above fig1 deals the overview of our approach where we consider both original and modified source codes and they can be preprocessed. It means to handle PHP web applications dynamic aspects and also to maintain input values in different versions because to hold all applications. We use a compiler called PHP compiler. Then about generation of test cases it contains test paths and where has constrain test paths and where has constrain value have usable and unusable and in this we consider unusable cases as reusing. Total information of test cases that can be analyzed and updated in reusable constrains. It analyses overall information available in test cases which are to be reused. The reusing test case contains information of the paths and what are the cases can be reused. This all information was available in constrains reusable. Finally, we have overall execution of test cases.

A. Collection of input values to be reused

It is difficult to generate test cases for input values to be reused. Mainly, preprocessing should be done then only generation of test cases can be processed. In preprocessing analysis the applications done in PHP format which can be controlled and passed through the compiler. The original and modified codes generate test cases from test paths because it is used to collect all the input variables together. Mostly, automatically tool may or may not check all the paths so in that case it should be done in manual way. Finally, reuse of constrains depends on complexity of recovery.

B. Algorithm to find input values to be reused

“Algorithm: Find Reusable Input Values

1. **Input:** old Path, new Path, old PDG, new PDG
2. **Output:** reusable variables
3. New PathVarDefUseMap: mapping variables to DefUse statement for the new path
4. For (n<-0){
5. n<newPath.size(),n++,do
6. currentBlock<-newPDG.getBlock(newPath.getBlockID(n))
7. If currentBlock.HasDefOrUse(){
8. UpdateDUM(currentBlock,newPathVarDefUseMap)
9. }}
10. oldPathVarDefUseMap: mapping variable to DefUse statements for the old path
11. for(n<-0){
12. n<oldPath.size()
13. n++,do
14. currentBlock<-oldPDG.getBlock(oldPath.getBlockID(n))
15. if(currentBlock.HasDefOrUse()){
16. UpdateDUM(currentBlock,oldPathVarDefUseMap)
17. }}
18. reuseVars: list of variables whose input values can be reused
19. for(n<-0){
20. newPathVarDefUseMap.getVars.Size(),n++,do
21. curVar<-newPathVarDefUseMap.getVars()[n]

```

22. if
    (oldPathVarDefUseMap.getVars().contains(curVar)){
23. Is sim<-
    chksim(curVar,oldPathVarDefUseMap,
    newPathVarDefUseMap)
24. If(is sim){
25. reuseVars.Add(curVar)
26. }}}}
27. Return reuseVars”

```

C. Explanation

Above algorithm takes inputs like old and new paths, old and new dependency graphs for program. The formal process of algorithm is to find test cases which are satisfying actual outputs and to reuse them. For mapping variables a new path is set and the program dependency graphs are listed out and also identifying new paths this was done in line3. To find the id's for block to path size and also each new path can be loaded into program dependence graph nodes in line7. Update def use map by considering variables of new path and current block in 10th line. From old path the variables are mapped to def use and also increment of path size is done.

Now the further step is to reuse input values and also make list of variables. Def use map in new version program defines nodes and they are extracted from source code. The def use map in older version finds variables to make source code to be extracted. Finally, the code statements in both versions are compared. This was done in different algorithm and both are combined later and to be found match between paths.

Mainly, the variables are to be same but condition used is different. For example, if the statement “if (b>=5)” is changed as “if (b>=5&& a<=1)”. The above first statement has been taken from modified version. Here the variable ‘b’ contains values greater than and equal to 5 but we are considering same variable in new version and also we include new variable ‘a’ and it shows values less than and equal to 1 means it can also display negative values. The only difference is variable ‘b’ is same in both versions but condition used is here different.

D. Module Description

The entire process of work can be subdivided into modules and can be explained in detail. The following modules of work are:

- Creating a website
- Assigning a database
- Testing the database
- Testing front end
- Performance evaluation

Creating a website:

Our paper shows the difference between old version and new version of the website. While change in features according to the requirement of users. Some test cases in the old versions may become useless and we may ignore them. Instead of deleting these cases we can reuse and can be used in modified version. First we should develop a website which is more useful for users. For example, we create a website called stocks of books gallery.

Assigning a database:

For creating a website we have shown the entire view of process and contents present in it. Now we need to assign a database which is created by using mysql, apache and so on. In database we include the details of customer, product, performance and many things can be added in future also. For example, in customer table it contains customer id, date of purchasing goods and complete details can be kept in tabular form and so on.

Testing the database:

After assigning the database there is need to test the database because we have to be accurate in designing of database. If any of requirements are missing then entire process will be waste. This is also used mainly where the change is required only that part can be retrieved.

Testing of frontend:

Now, we can test the front end process. This can be done by giving the input values or constraints and we check the exact value is displayed or not. This testing is mainly done because user can test the product and satisfy by showing the exact results of required output. If

not then the data can be modified in the back end and can be display in front end.

Performance evaluation:

Then finally, after completion of all the above steps then main important process which is used to say about the overall view of the process is evaluation of process. It can be declared by comparing of old and new versions of process. This can be displayed in the form of graphs, charts, pictures, tabular formats and so on.

4. DATA AND ANALYSIS

In this part we discuss about content of data and analyze them. Suppose, we are considering a web page of books gallery containing customer, orders, shipping, and many more. According to requirements of particular website we should design a data flow which shows overall structure of process. While collecting data we should fix bugs and they should not proceed in modified. Then only the flow of data can reach expected output. Mainly, it should focus on overall flow whether satisfying conditions or not. The data flow is represented below.

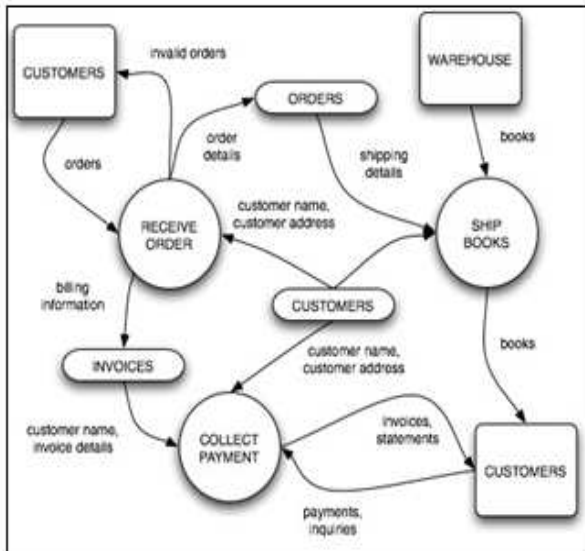
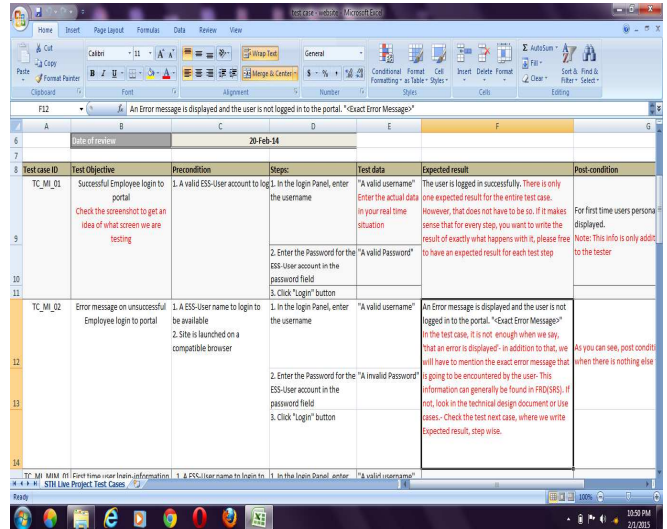


Fig 2: Flow of Data

in fig 2 we show how data flows and the way it proves to reach expected output. It mainly consists of details of customer like they use to order and receive them. While the agent of

supplier collects all information from customers to maintain perfect software. In this we can show some of requirements included and satisfied more in modified version. So, such cases can be reused in latest version by making minor modifications like changes in input constrains. So, testing can be done with less effort by selecting the paths where modification occurred.



test case ID	Test Objective	Precondition	Steps	Test data	Expected result	Post-condition
TC_M1_01	Successful Employee login to portal Check the screenshot to get an idea of what screen we are testing	1. A valid ESS-user account to login to the portal	1. In the login Panel, enter the username 2. Enter the Password for the ESS-user account in the password field 3. Click "Login" button	"A valid username" "A valid Password"	The user is logged in successfully. There is only one expected result for the entire test case. However, that does not have to be so. If it makes sense that for every step, you want to write the result of exactly what happens with it, please free to have an expected result for each test step	For first time users persona Note: This info is only add to the tester
TC_M1_02	Error message on unsuccessful Employee login to portal	1. A ESS-user name to login to be available 2. Site is launched on a compatible browser	1. In the login Panel, enter the username 2. Enter the Password for the ESS-user account in the password field 3. Click "Login" button	"A valid username" "A invalid Password"	An Error message is displayed and the user is not logged in to the portal. "Exact Error Message" in the test case, it is not enough when we say, "that an error is displayed"- in addition to that, we will have to mention the exact error message that is going to be encountered by the user. This information can generally be found in FREQSRS, if not, look in the technical design document or use cases- Check the test next case, where we write Expected result, step wise.	As you can see, post conditio when there is nothing else

Table I. Test Cases Template

The above template contains details about test cases developed in our application and this template is sample part of that. Actually, it has several columns and rows where it has total details of software. It has conditions before testing and after testing where it contains test case id here it shows in auto increment manner. It shows expected output of particular test cases and some steps are involved to fulfill condition then data is entered successfully by covering path coverage. Finally, it reaches the condition and makes test cases successful or else it shows failure.

5. DISCUSSION AND RESULT

By using regression testing our approach proves whether it can generate efficient test cases or not. We can compare results by considering input values by taking usable and unusable test paths. So, we collect data in three forms they are: (1) how many reusable input values are there and list them. (2) Input constrains to be required and should execute new test paths. (3) Input values percentage to be listed.



The modified versions also considered to make above forms to conform the list of test paths needed to be reused. Affected test paths to be selected and on particular affected area should be retested by using regression testing. Gather over all percentage of reused constrains of both versions and make a list. For each new application it is mentioned in increment order of versions because old versions have already completed regression testing.

Mainly we concentrated on web application because here number of changes will be updated frequently. Main aim is to reuse the test path by using different values without changing constrains. We mostly concentrated on part where code is affected and testing is done in this manner. Many automatic tools are available to do testing. Suppose, the tool fails to check all test cases then we need to do in manual way to complete total work. By using manually we should use more effort to fulfill our desire.

“n0.php (original version)

```

1. $a= $_POST['input 1']
2. $b= $_POST['input 2']
3. If($k<10){
4.   $m=5;
5.   $k=$k-4;}
6. $m= $m+6;
7. If($k>7){
8.   If($m==5)
9.     echo “k\n”;
10. else
11. $m=7;}
12. echo “m\n”;
13. echo “process is done\n”;”
    
```

Simple PHP program is considered and changes made to prove the reusing of values. Example program is the old version (n0.php). Here, the modified paths are considered and checks whether expected output is made or not.

“n1.php (modified version)

```

1. $a=$_POST['input 1'];
2. $b=$_POST['input 2'];
3. If($k<10){
4.   $m=$m-1; // modified
5.   $k=$k-2;}
6. $m=$m+6;
7. If($k>7){
8.   If($m==5)
9.     echo “k\n”;
10. else
    
```

```

11. $m=7;}
12. echo “m\n”;
13. echo “process is done\n”;”
    
```

Modified program (n1.php) shows the changes done in old version. We can observe in line4 modification is done as (\$m=5) changed into (\$m=\$m-1). So when the change is made then increase of test paths also occurs. We can also observe changes occurred in variables ‘k’ and ‘m’. The test paths vary in both new and old versions.

n0.php	n1.php
\$ k<10	\$ k<10
\$ k-4>7	\$ k-4>7
\$ m==5	\$ m-1==5

Table 2: Input Values

In above table1 we can see the changes made to show how the input values are differing. Constrains are same but the conditions applied here are different. By definition graph we can collect the changes made in input values and also we can take by comparing with old test paths. We observe in the versions used variables ‘k’ is neutral so we can reuse this in new version by having different input values. But in the variables ‘m’ we can see changes in conditions made to reuse in both versions. So, it can make differs in test paths and also can be seen in definition graphs used in programs. By this finally, we can find the test paths that can be reused and make regression testing cost to decrease to make less effort.

6. CONCLUSION AND FUTURE WORK

Finally, reuse of unusable test cases can be done in our approach. In modified version instead of creating new test case and make work more complicate we reuse the test cases by having input constrains. We cannot assure of complete reuse but maximum can be reused in new versions. For GUI applications this approach can be more useful to make less effort on frequent changes. In future many technologies can be implementing by using this ideology. It can show partial results of reusing test paths. By using regression testing our approach can be effective and time complexity decreases. In future, complete test cases can be reused by

making only changes in input constrain and make applications more effective.

REFERENCES

- [1] Aaran marback, Hyunsook Do, Nathan Ehresmann, "An effective regression testing approach for PHP web applications-IEEE fifth International Conference on software testing, verification and validation,(2012), Page: 312 - 321".
- [2] B. Bharathi, "A simple method for deriving LQN models from software models represented as UML diagram – Indian Journal of Science and Technology, vol5, Issue-2, (2012), Page: 2148 to 2154".
- [3] B. Bharathi, "Step by Step Approach to Convert Software Models Represent as UML Diagrams with SPT Profile to LQN Performance Models – National Journal on Advance in Computing and Management , (2012), vol2, no-2, Page: 6 to 10".
- [4] Hung vietnguyen, Hoan ANH Nguyen, Tung Thanh Nguyen, Tien N. Nguyen "DRC : A detection tool for dangling references in PHP – based web application – ICSE 2013, Sanfransico, CA,USA formal demonstration, Page: 1299 - 1302".
- [5] Ivan Enderlin, Alain Giorgetti, Fabrice Bouquet, "A constrain solves for PHP arrays – IEEE sixthinternational conference on software testing, verification and validation workshops, (2013), Page: 218 - 223".
- [6] L.C. Briand, Y. Labiche, G. Soccar, "Automating impact analysis and regression test selection based on UML designs – IEEE international conference on software maintenance (1CSM'02), (2002),".
- [7] Md. Hossain, Hyunsook Do, Ravi Eda, "Regression Testing for Web Applications Using Reusable Constraint Values – IEEE International conference on software testing, verification, and validation workshops, (2014),".
- [8] Marcio geovani jasinski, Luis Bernardes, Carla Diacui Medeiros Berkenbrock, "A case study of natura campus open innovation platform – a collaborative system overview – proceedings of IEEE 18th international conference on computer supported cooperative work in design,(2014), Page: 446 - 450".
- [9] Nam ye, xinchen, Peng Jiang, Wenxu Ding, Xuandong Li, "Automatic regression test selection based on activity diagrams – fifth international conference on secure software integration and reliability improvement companion, (2011), Page: 166 - 171".
- [10] Nai-lun huangl, Tsern-Huei Lee, "Solution of skip distance constraint on sub - linear time string – matching architecture – IEEE conference, (2013),".
- [11] Nenad ristic, mladen veinovic, aleksandar jevremovic, "Improving protection of PHP source code using cryptology models – Telskis, NIS, Serbia, October 16-19, (2013) , Page: 409 - 412".
- [12] Orestpilskalns, gunayuyan and anneliese Andrews, "Regression testing UML designs – 22nd IEEE international conferences on software maintenance (ICSM'06), (2006),".
- [13] Robert feldt, Greger Wikstrand, Jeevan Kumar Gorantla, "Dynamic regression test selection based on a file cache – an industrial evaluation – International conference on software testing verification and validation,(2009), Page: 299 - 302".
- [14] Sebastian elbaum, Alexey G. Malishevsky, Gregg Rothermel, "Test case prioritization: A family of empirical studies - IEEE transactions on software Engineering, vol28, no.2, February 2002, Page: 159 - 182".
- [15] Suman bhowmick, Biswarup Das, Narendra Kumar, "An advanced IPFC model to reuse Newton power flow code – IEEE transaction on power system, (2009), vol-24, issue no.20, Page: 525 - 532".
- [16] Walid said abdel – hamid, Mohamed hadhoud, "Regression test selection technique based on dynamic behavior – IEEE conference volume 3,(2010), Page: 346 - 350".
- [17] Yannick L.H.Llew yaw fung, Anjaneyulu Pasala, Fady Akladios, Appala Raju G, Ravi P Gorthi, "Selection of regression test suite to validate software application upon deployment of upgrades – 19th Australian conference on software engineering,(2008), Page: 130 - 138".