International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 15 (2018) pp. 11776-11780

Test Case Prioritization Using Requirements Clustering

Rayapureddy Kalyani [1], Padmanabhuni Sai Mounika [2], Ravipati Naveen [3], Gnaneswari Maridu [4]

[1,2,3,4]Students, Department of Information Technology,
VR Siddhartha Engineering, College, Kanuru, Vijayawada, Andhra Pradesh, India.

Paruchuri Ramya[5]

[5]Assistant Professor, Department of Information Technology,
VR Siddhartha Engineering College, Kanuru, Vijayawada, Andhra Pradesh, India.

Abstract

Regression testing is the procedure of retesting the product and checking whether additional faults or errors have been created in the existing one. It is vital for keeping up programming quality. But it is a costly process. By, utilizing prioritization technique we can diminish the cost. Prioritization increases productiveness of regression testing and its main criteria is to build the rate of error detection. Merging requirements information into current testing practice helps the engineers to recognize the source of faults easily. In this paper we research whether the requirements based grouping methodology can enhance the viability of prioritization techniques. So, here we perform grouping approach on given requirements and prioritization techniques based on code scope metric.

Keywords: Regression testing, Test case prioritization-necessities based grouping, Code scope metric.

INTRODUCTION

Software testing is a procedure of evaluating the mistakes in an application. It helps to recognize mistakes, holes, missing necessities to that of real prerequisites. It should be possible by either manual testing or robotized instruments. Some favour saying programming testing as clear box and behavioural testing. There are two sorts of programming testing, which is performed in programming advancement life cycle. Manual/Behavioural Testing is the process of testing the defects present in software or an application. This testing is performed manually without using any tool and it is mainly performed by humans. To know the completeness of testing, testers uses test cases, test plans to test the software. On the other hand, tester writes scripts in Automation Testing to test the software. This is done mainly by using an automated tool. The tool helps us to execute the entire test-suites and capable of executing the tests. Test automation increases the accuracy, test scope, spares money when compared to manual testing.

When the software is tested manually or automatically, we need to perform the Regression testing for every new release of the software. This is a standout amongst the most helpful programming testing amid programming upkeep. Regression/Relapse implies procedure of testing programming to ensure that more established despite everything one works with new changes. It is essential for large organization. It is handled by testing apparatuses for human testers to perform similar task.

Benefits of Regression testing

- It doesn't show any impact on the work that has been verified.
- It is done during integration testing and it is more useful.
- In very less time it increases the test coverage
- It is cost effective.
- It ensures high quality software by executing the steps repeatedly.
- It makes sure that defects created are fixed or not.

Techniques of Regression Testing

The systems of regression testing are as per the following:

- Retest all is a costly procedure and includes gigantic time and more resources. The total test present in the collection of test cases is re-executed and it checks it integrity by checking the test cases in the current program.
- In Regression test selection a part of test suite is only selected for the execution part[6]. The testcases present here are divided into reusable experiments and experiments. The proceeding regression cycles use the reusable experiments and the obsolete experiments are not utilized as part of succeeding lifecycles.
- Prioritization of experiments mostly relies upon the business affect and mostly part utilized functionalities. It will probably build the capacity to meet execution objectives like rate of blame recognition, rate of scope. The high need test cases will help in lessening the relapse test suite

Machine Learning

Machine learning is the execution of computerized reasoning for automating the system and to reduce the burden on

human. Now-a-days, it is getting more popular due to its growing demand. It is divided into three types:

1. Supervised learning: It consists of target which is predicted from given set of inputs. Example: Decision tree
2. Unsupervised learning: Here we don't have any objective. It is utilized for clustering Example: K-means
3. Reinforcement learning: In this the machine is prepared to learn by exposing it to some environment.
4. Example: Markov decision process

PROBLEM STATEMENT

Regression testing is very critical for maintaining programming quality and it is very expensive and takes much time to execute the test cases. Automation testing is not alone enough because we cannot make changes whenever we want. So, we have to wait until all the process is done. If regression testing is done without automated tools then it will be more time consuming. So, we will do both manual testing and automation testing also. Utilizing the requirements can be more helpful to prioritize the test cases as per the requirements could potentially help to find out the problems in these cases. The importance of combining requirements during testing has been surely known by necessities designing group. So, in this research we explore whether grouping experiments in view of necessities may enhance the adequacy of regression systems.

LITERATURE REVIEW

It is suggested that another prioritization technique using Genetic Algorithm (GA)[4]. This method examines the hereditary calculation as to adequacy time overhead by using fundamentally based measure to organize test cases. It also organizes the test suite so that additional test suite will have an unrivalled rate of error recognition. A novel thickness based k-means groping algorithm has proposed to conquer the disadvantages of DBSCAN and k-means grouping algorithm gives an enhanced variant of k-means clustering algorithm[5]. This algorithm will perform superior to DBSCAN. An imaginative genetic algorithm to organize the regression test suite is presented that will organize test instance of average percentage of fault detection (APFD) measure. Another prioritization procedure for grouping way to enhance prioritization and for genuine flaws, use code scope and code intricacy. It was also proposed that grouping/clustering based approach and carries their performance with average percentage of fault detection (APFD) measure. The necessities based grouping approach combines the code analysis can enhance the adequacy of prioritizing the test cases. Another procedure of analysing the level of experiments performed to discover the errors and on APFD metric's outcomes. To enhance the regression/relapse testing process, test case prioritization methods used to set up the execution of test cases.

PROPOSED METHODOLOGY

The principle layout is to portray a specialized way to deal with organizing the experiments in view of requirement based clustering, this approach comprises of following advances:

1. Requirements clustering
2. Mapping of necessities to test cases
3. Prioritizing the test cases
4. Clustering the prioritization
5. Test case selection from the clusters

Clustering the requirements

We use the three minor process to get this step done, based up on text mining we utilize term extraction, term-report lattice and k-means bunching

Term extraction: Here we consider the requirements from the user and words are extracted from these requirements and the words which do not have any meaning in these requirements such as prepositions, articles are removed or not considered. All the distinct words are considered and used for next process.

Term-document matrix: we use the words obtained from above step and make term-document matrix. This is simply a matrix of words in which the rows corresponds to requirements and columns correspond to distinct terms. Matrix can be of many ways as we wanted.

K-means clustering: This is the important process in the step, there are many clustering algorithms but we use k means clustering algorithm because its proved that it is good for document clustering and many other factors such as simple and cost effective to implement[5].

This allows to take specific number of clusters, in this calculation for n number of necessities and p number of terms k implies calculation assigns every prerequisite to bunches to limit bury group total as takes after.

$$\text{Sum}(k) = \text{sigma}(x(i, j) - x(k, j))^2$$

where x (k , j)= mean variable, j of all components in k.

Initially stage is to decide k focuses aimlessly one for each cluster. Next stage is to decide the separation between information focuses to its closest cluster. To ascertain remove we utilize Euclidean separation formula. Initial gathering is done when every one of the information focuses are incorporated into a group. If any new data points we update the centres, until no new centres are found.

Mapping requirements to test cases

Here once we obtained the cluster of requirements we use the matrix of requirements-test case in this process[13]. The

requirement-tests mapping resolution process by perusing the prerequisites in the groups and recognize their comparing experiments from the framework. By the comparison of the words present in the requirement and test cases are done and then if the word obtained is in any particular cluster we map the requirement to the test case. And the similar words are taken and added to the experiments.

Prioritizing the tests

After the groups are made we organize the experiments in the bunches utilizing prioritization techniques, here we utilize code intricacy as prioritization technique. To ascertain the code many-sided quality we utilize 3 sorts of data from source code they are characterized as:

- Lines of code: It measures the aggregate number of lines in the code, we consider just executable code and disregard remarks and clear lines.
- Settled square depth: It measures the quantity of settled pieces in the announcements.
- Cyclomatic complexity: It measures the quantity of directly freeways by examining the choice tree structure of a strategy.

Clustering the prioritization

Every one of the necessities are not critical to the clients, certain programming prerequisites related are more used than the other so certain prerequisites needs more consideration from the analysers. So we organize the prerequisite bunches with the goal that we can use their need necessity when each experiment is chosen.

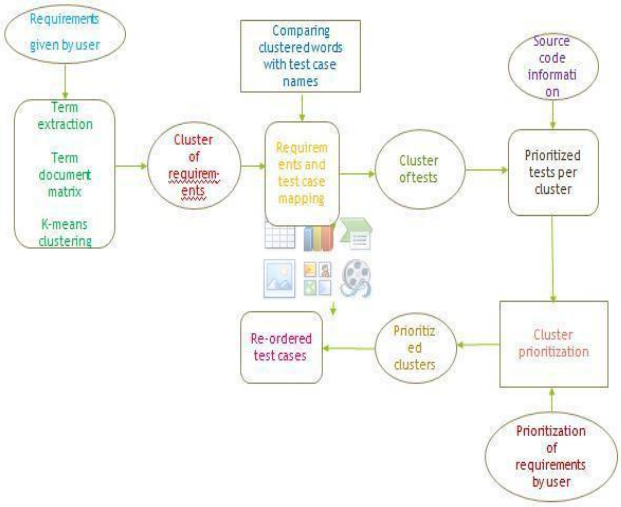
A few organizations regularly take the necessities and organize them and convey the item as per them. To organize the groups we utilize the dedication level and we utilize code change data that can be great pointer for the nearness of faults. Using these two we organize the bunches.

Test case choice

When we have organized experiments for each cluster, we need to make an entire arrangement of recorded experiments in each cluster, to do as such we have three strategies:

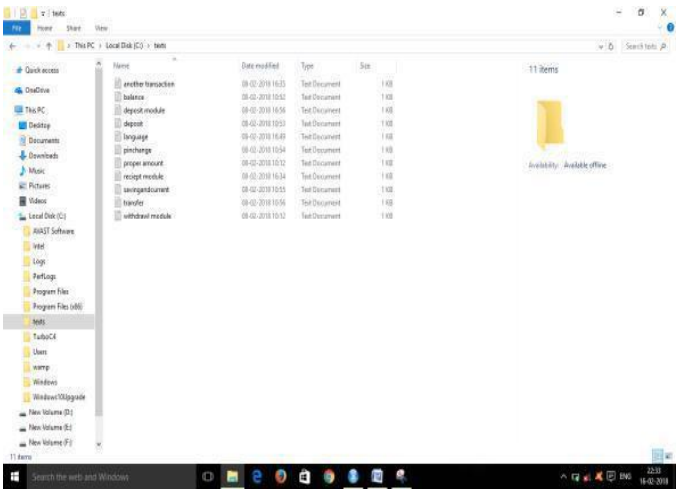
- Unique request of clusters: This technique visit the groups in the request produced by the bunching tool, pick the principal experiment in the group and pick the second and rehash a similar utilizing round robin strategy until the point when each experiment is picked.
- Arbitrary request of clusters: This technique visits the groups in irregular request and applies an indistinguishable strategy from initial one.
- Organized request of clusters: This technique visits the bunches in organized order, when we are picking the experiments from the clusters, we select the experiments which are more need than the other test cases, to do as such we ascertain normal number of experiments per cluster. Then we take t number of experiments from the most astounding positioned cluster. And the quantity of experiments chose will go down to 10% of t for ensuing clusters; the process is rehashed until all the experiments have been picked.

ARCHITECTURE DIAGRAM

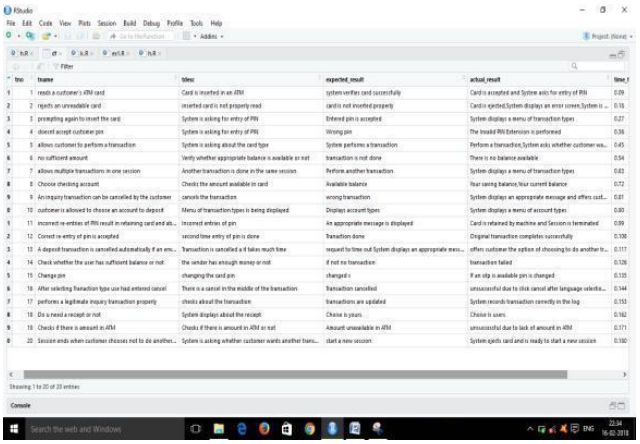


RESULTS AND OBSERVATIONS

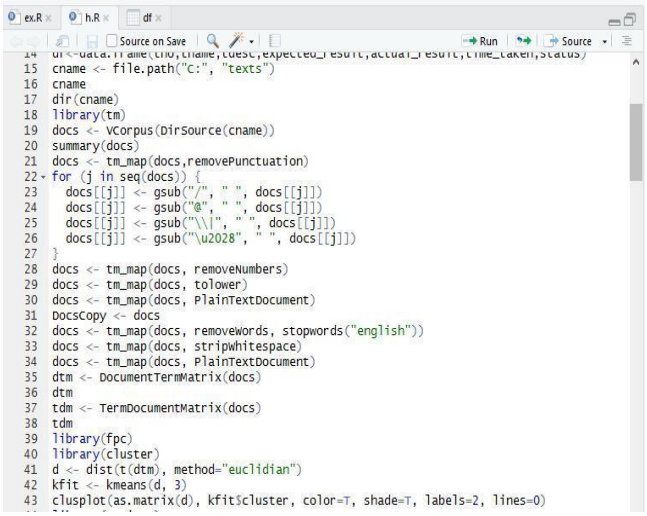
The figure shows the list of documents we have considered for atm application like deposit module, withdrawal module, pin change module etc. We have considered 11 requirements specification documents



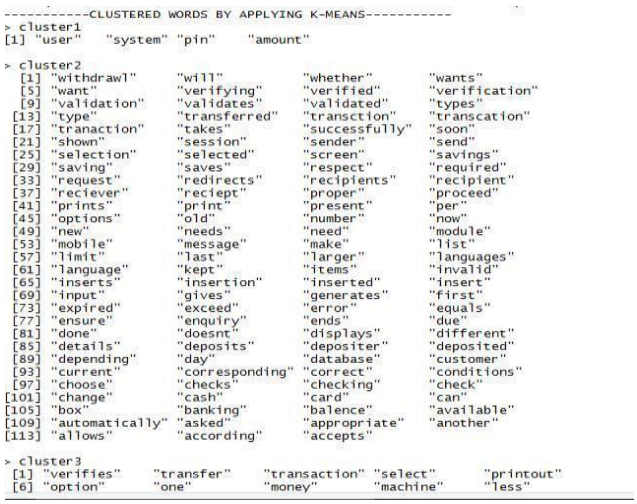
Now we have written test cases for this application manually in R studio. The test case template includes test case id, test case name, test case description, expected result, actual result, time take and status.



Now we applied some pre-processing that means text mining on the given documents. This include removing stop words in English, extra spaces, numbers, special characters etc.



```
cname <- file.path("C:", "texts")
cname
dir(cname)
library(tm)
docs <- VCorpus(DirSource(cname))
summary(docs)
docs <- tm_map(docs,removePunctuation)
for (j in seq(docs)) {
  docs[[j]] <- gsub("/", " ", docs[[j]])
  docs[[j]] <- gsub("@", " ", docs[[j]])
  docs[[j]] <- gsub("\\|", " ", docs[[j]])
  docs[[j]] <- gsub("\u2028", " ", docs[[j]])
}
docs <- tm_map(docs, removeNumbers)
docs <- tm_map(docs, tolower)
docs <- tm_map(docs, PlainTextDocument)
DocsCopy <- docs
docs <- tm_map(docs, removeWords, stopwords("english"))
docs <- tm_map(docs, stripWhitespace)
docs <- tm_map(docs, PlainTextDocument)
dtm <- DocumentTermMatrix(docs)
dtm
tdm <- TermDocumentMatrix(docs)
tdm
library(fpc)
library(cluster)
d <- dist(t(dtm), method="euclidian")
kfit <- kmeans(d, 3)
clusplot(as.matrix(d), kfit$cluster, color=T, shade=T, labels=2, lines=0)
```



```
----------CLUSTERED WORDS BY APPLYING K-MEANS----------
> cluster1
[1] "user"    "system"  "pin"     "amount"

> cluster2
  [1] "withdrawl"     "will"          "whether"       "wants"
  [5] "want"          "verifying"     "verified"      "verification"
  [9] "validation"    "validates"     "validated"     "types"
 [13] "type"          "transferred"   "transction"    "transcation"
 [17] "tranaction"    "takes"         "successfully"  "soon"
 [21] "shown"         "session"       "sender"        "send"
 [25] "selection"     "selected"      "screen"        "savings"
 [29] "saving"        "saves"         "respect"       "required"
 [33] "request"       "redirects"     "recipients"    "recipient"
 [37] "reciever"      "reciept"       "proper"        "proceed"
 [41] "prints"        "print"         "present"       "per"
 [45] "options"       "old"           "number"        "now"
 [49] "new"           "needs"         "need"          "module"
 [53] "mobile"        "message"       "make"          "list"
 [57] "limit"         "last"          "larger"        "languages"
 [61] "language"      "kept"          "items"         "invalid"
 [65] "inserts"       "insertion"     "inserted"      "insert"
 [69] "input"         "gives"         "generates"     "first"
 [73] "expired"       "exceed"        "error"         "equals"
 [77] "ensure"        "enquiry"       "ends"          "due"
 [81] "done"          "doesnt"        "displays"      "different"
 [85] "details"       "deposits"      "depositer"     "deposited"
 [89] "depending"     "day"           "database"      "customer"
 [93] "current"       "corresponding" "correct"       "conditions"
 [97] "choose"        "checks"        "checking"      "check"
[101] "change"        "cash"          "card"          "can"
[105] "box"           "banking"       "balence"       "available"
[109] "automatically" "asked"         "appropriate"   "another"
[113] "allows"        "according"     "accepts"

> cluster3
 [1] "verifies"   "transfer"   "transaction" "select"     "printout"
 [6] "option"     "one"        "money"       "machine"    "less"
```

In this step we have mapped the clustered words with test case names. Mapping is noting but checking the word matching with test case name and classify the test cases according to the word clusters

```
----------AFTER MAPPING TESTCASE IDS----------
> cc1
[1] 11 20 14  2 10  4  1

> cc2
[1] 11

> cc3
 [1]  7  8  9 15  5 16 19  6 13 11 12
>
```

Now we prioritized the test cases based on code complexity metric. This is done by summing up the values of lines of code, nested block depth and cyclomatic complexity. The test case with high value must be tested first because it has more complexity. Thus, the prioritization takes place within each cluster itself. The resultant test case ID's after cluster prioritization are:

Now we apply k-means algorithm on the pre-processed data. The figure below depicts plotted clusters. We need visual representation for better understanding the k-means result.

```
[1] "THE PRIORITIZED TESTCASE IDS AFTER CLUSTER PRIORITIZATION ARE"

> res
 [1]  4 19 16  6 12 15  9  7 20 11 13 17  5 18  3 10  2 14  1  8
>
```

CONCLUSION

In this study we found that incorporating requirements in testing phase has helped a lot in finding the faults and errors easily. Although there are many prioritization techniques , without using source code information the effectiveness is not up to the mark. The percentage of productivity has been increased to 80% by using requirements information in prioritizing the test cases.

REFERENCES

[1] Srivastava, Praveen Ranjan, "Test case prioritization", Journal of Theoretical and Applied Information Technology, 2008.

[2] 2.Liela Naslavsky, Hadar Ziv, Debra J. Richardson, "A Model Based Regression Test Selection Technique", International Conference on Software Maintenance (ICSM), IEEE, 2009.

[3] 3.Yamini Pathania, Gurpreet Kaur, "Role of Test Case Prioritization Based on Regression Testing Using clustering, 2015.

[4] 4.Krishnamoorthi, S. A Sahaaya and Arul Mary, "Regression Test Suite Prioritization Using Genetic Algorithm", International Journal of Hybrid Information Technology, 2009.

[5] K. Mumtaz and Dr K. Duraiswamy, "A Novel density based improved k-means clustering

algorithm-Dbkmeans", International Journal on computer science and Engineering, 2010.

[6] Supriya S. Lichade, Praveen Thakur, "Test Case Prioritization Using Regression Testing",2016.

[7] Arvinder Kaur and Shubhra Goyal, "A genetic algorithm for regression test case prioritization using code coverage" International journal on computer science and engineering, 2011.

[8] Arafeen, Md Junaid and Hyunsook Do, "Test case prioritization using requirements-based clustering", IEEE sixth international conference on software testing, Verification and validation, 2013.

[9] Carlson, Ryan, Hyunsook Do, and Anne Denton, "A clustering approach to increase test case prioritization: An industrial case study", 27th IEE international conference on software maintenance (ICSM), 2011.

[10] 10. Muthusamy, Thillaikarasi, "A new effective test case prioritization for regression testing based on prioritization algorithm", International Journal of Applied, 2014.

[11] 11. Hettiarachchi, Charitha, Hyunsook Do, and Byoungju Choi, "Effective regression testing using requirements and risks", 2014.

[12] 12. Medhun Hasini D.R, "Clustering approach to test case prioritization using code coverage metric" International Journal of Engineering and Computer Science, 2014.

[13] 13. A.K Misra, "Prioritizing test suites using clustering approach in software testing", International Journal of soft computing and engineering, 2012.