JCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Volume-5, Issue-7 E-ISSN: 2347-2693

A Novel Approach for TCP for Regression Testing In Web Application

Priyanka P Deshmukh[1*], **Vina M Lomte**[2]

[1*]Dept. of Computer Engineering, RMD Sinhgad School of Engineering, Pune, India
[2]Dept. of Computer Engineering, RMD Sinhgad School of Engineering, Pune, India

**Corresponding Author:* *priya.atf@gmail.com, vina.lomte.rmdssoe@sinhgad.edu*

Available online at: www.ijcseonline.org

Received: 21/May/2017, Revised: 09/Jun/2017, Accepted: 28/Jun/2017, Published: 30/Jul/2017

Abstract- For increase rate of fault detection test case prioritization is needed, which shows how fast bugs are identified during the testing phase. In test case prioritization follow way to use the information of previously executed test cases, such as coverage information, resulting in an iterative prioritization algorithm. Real fact of using coverage information can improve the rate of fault detection in prioritization algorithms. But performance of such iterative prioritization schemes degrade as the number of ties occurred in prioritization steps increases. In test case prioritization using lexicographical ordering and extended diagraph, we propose a new heuristic for breaking ties in coverage based techniques. Performance of the proposed technique in terms of the rate of fault detection is comparatively evaluated using a wide range of programs. Results indicate that the proposed technique can resolve ties and in turn noticeably increases the rate of fault detection.

Keywords- Additional statement coverage, Fault-based test case prioritization, GUI testing, HMM, model-based testing (MBT), random prioritization, reinforcement learning.

I. INTRODUCTION

The complexity and size of software systems are growing, along with the increasing importance of testing and verifying these systems. As a result, many test suites produced during development are reused in a regression testing mode, especially during software maintenance or evolution. A software product, once developed, has a long life and evolves through numerous additions and modifications based on its faults, changes of user requirements, changes of environments, and so forth. With the evolution of a software product, assuring its quality is becoming more difficult because of numerous release versions [1]. Users expect to get a new and better quality software version than before. In some cases, the quality of software becomes worse than before because of the added or modified features which create additional faults into the existing product as well as the newly modified version. For assuring good quality software, testing is mandatory. Evaluating a system with the intention of finding faults is known as Software Testing. Once system has been developed, it must be tested before implementation. It is oriented towards Error-detection [2]. Software testing is one of the major and primary techniques for achieving high quality software. It is done to detect the presence of faults, which cause software failure. It can also be referred as the process of verifying and validating software application or program to ensure that software meets the technical as well as business requirements as expected [3-17].

a. Background

For testing, a software engineer often uses test cases. A test case is a set of conditions or variables and inputs that are developed for a particular goal or objective to be achieved on a certain application to judge its capabilities or features. It might take more than one test case to determine the true functionality of the application being tested. Every requirement or objective to be achieved needs at least one test case. Some software development methodologies like Rational Unified Process (RUP) recommend creating at least two test cases for each requirement or objective; one for performing testing through positive perspective and the other through negative perspective. Regression testing is a kind of software testing that focuses on selective retesting through various versions of a software system [5]. The following is the formal definition of regression testing used by IEEE. “Selective retesting of a system or component to verify that modifications have not caused unintended effects and that the system or component still complies with its specified requirement.”[6] Another popular software testing technique is Test Case Prioritization. In this technique, each test case are assigned a priority. Priority is set according to specific criterion and test cases with highest priority are scheduled first. Another criterion may be the rate at which fault is detected. [7] The goal of this research is to find a metric to quantify the rate of dependency detection among faults and provide an algorithm that prioritizes the test cases in an order that has improved dependency detection rate compared to non-prioritized test cases. By the definition of the test case prioritization, problem represents a quantification of such goals.

b. Motivation

In the paper [8], researchers proposed an algorithm to measure effectiveness of test case prioritization in regression testing and a prioritization technique which can be used to improve the fault detection process for regression testing. In [8], researchers only considered the dependent faults which are fully dependent on other leading faults. But did not consider the fact that, there can be faults that are not fully dependent rather mutually dependent on more than one fault. The detection of the independent and fully dependent faults is covered simultaneously in this software testing approach. But, an efficient example needed to be set along with the independent and dependent faults (both fully & partially) to make an efficient approach. Further, a sizable performance gap can be seen as prioritization is done only with taking the fully dependent faults into consideration, not the partially dependent faults.

Hence, in order to overcome these issues, in current research paper, we will extend this research work to investigate the above mentioned weaknesses and will provide an alternative or improved version of prioritization technique including different methods of fault detection methods. A thorough research in this field may help to detect faults as early as possible.

In this paper, we will extend the research of prioritizing test cases considering fault dependency mentioned in [8], as we think fault dependency consideration is incomplete there. In this paper, our goal is to include the fault dependency considering both fully & mutually dependent faults for doing complete test case prioritization.

The section I includes Introduction which describes the background & motivation of this work. Section II presents the relevant work published in literature in the area of test case prioritization. Section III details about mathematical model. Proposed system is presented in section IV. The results are discussed in section V. Section VI presents the Conclusion.

II. LITERATURE REVIEW

This paper proposed an approach for test case prioritization in order to improve regression testing. Analysis is done for prioritized and non-prioritized cases with the help of APFD (average Percentage fault detection) metric. It is proven that when the prioritized cases are run then result is more efficient. In future test case prioritization can be done by using more factors and evaluate By PTR and risk metrics [1]. In this paper we describe requirement based test case prioritization technique. This proposed Technique is highly useful to identity and evaluate various issues arises while working with varying requirement environment. The proposed prioritization technique used most efficient Factors to prioritize test suite because the errors introduced in the requirement phase is approximately 50% of all faults detected in the entire project. The change in requirements is the major factor attributable to the failure of the project so we prioritize test cases according to Requirement priority and requirement factors [2].

This paper deals with ESG model based test case prioritization problem for a large number of test cases. Improving our previous study, new model-event based TCP approach where instead of ordering indirectly test cases according to their preference degree they are automatically divided into the five groups (classes). It is provided thanks to representing prioritization group label of each test case as output depending on two attributes: important index weighted by membership degree and frequency of occurrence of all events belonging to given group. Then, such a way for all test cases (100) formed data set is classified by using MLP neural network. The structure of NN-classifier for commercial test application has been defined and its performance is examined. Application results show high classification accuracy and the acceptable test prioritization performance [3].

Our algorithm is based on analysis of the percentage of test cases performed to find the faults and on APFD metric's results. Abiding by the percentage of executing test cases in earlier fault detection is important as sometimes regression testing ends without executing all test instances. Outcomes demonstrate that our algorithms can also achieve better execution in this event. For instance, in the first project if only 75% test cases could be melt down due to resource constraint, random strategy could find more or less 66% faults; while our proposed algorithm detects about88% faults. In a second project if we consume 30% test cases to accomplish; then random strategy could find more or less 27% faults; while our proposed algorithm detects about 40% faults. This shows clear evidence that our proposed algorithm is a lot better in earlier fault detection than random technique. The graphical representation of these outcomes is presented at a lower place [4].

In this paper, we proposed a new prioritization technique for prioritizing system level test cases to improve the rate of fault detection for regression testing. Here we propose new practical set of weight factors used in the test case prioritization process. The new set of are tested for the regression test cases. The proposed prioritization algorithm is validated by using APFD metric. Experimental Results shows that proposed technique leads to improve the rate of fault detection in comparison with random ordered test cases and reserves the large number of high priority test with least total time during a prioritization process [5].

In this paper a new prioritization technique to improve the rate of fault detection of severe faults for Regression testing is proposed. Here, two factors rate of fault detection and fault impact for prioritizing test cases are proposed. The proposed algorithm is validated by analyzing two sets of industrial projects. Results indicate that the proposed technique lead to improved rate of detection of severe faults in comparison to random ordering of test cases. And also it is tested experimentally that the number of test cases runs to find the entire fault is less in case of proposed prioritization technique. The results prove that the proposed prioritization technique is effective. In future, test case prioritization over requirement analysis will be tried [6].

In this section we are going to discuss about Software testing. Then we will focus about the importance of software testing and test cases. Finally we will narrow down the topic into test case structures, test case designing and test cases. We will also focus on test case prioritization technique, existing techniques for test case prioritization, its problems and our focus area.

III. MATHEMATICAL MODEL

Let Us consider Test case prioritization and bug triage

S={s, e, X, Y, Fme, DD, NDD, Φ}

Where,

s = Start of the program.

1. Log in with System.
2. Load dataset of bugs for test cases.
3. Word dimension
4. Bug dimension

e = End of the program.

Test cases prioritization using Lexicographical Ordering and extended diagraph

X = Input of the program.

Test cases from sample bug report.

Y = Output of the program.

Reduced Bug report for developer to fix them

X, Y $\in$ U

Let U be the Set of System.

U= {User, FS, IS, Cluster}

Where User, FS, IS, Cluster are the elements of the set.

User=Developer

FS=Feature Selector

IS=Instance Selector

Cluster= Cluster

Fme = {F1, F2, F3, F4, F5}

Where,

F1= Input bug report and words.

F2= Apply CH->ICF and ICF ->CH

F3= Naïve Bays Algorithm .

F4= Genetic Algorithm.

F5= Generate reduced bug report.

$$P(s_{t+1|}s_{t-1,}s_{t-2,\dots\dots}) = P(s_{t+1|}\ s_t) = T(\ s_{t,}\ a_{t,}\ \ s_{t+1})$$

The value function $V^{\pi}\ (s)$, specifies “how good” it is for the agent to be in a given state. The how good notation here is expressed in terms of future rewards that can be expected. We can define the value of state s under policy π, formally, $V^{\pi}\ (s)$.

$$V^{\pi}\ (s) = E_{\pi}\ \{R_t\ \ |s_t = s\} = E_{\pi}\ \{\sum_{k=0}^{\infty} \gamma^k\ \ r_{t+k+1}\ |s_{t=s}\}$$

Similarly, the value of performing an action a in state s (the state action value function, or Q-value function, Q : S × A→ R) can be defined as

$$Q^{\pi}\ \ (s, a) = E_{\pi}\ \ \{R_t\ |s_t = s, a_t = a\} = E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k\, r_{t+k+1}|s_t = s, a_t = a\}$$

Q-learning estimates the agent’s Q-value function based upon an action’s Q-value estimation; this process is incrementally evaluated as follows

$$Q_{k+1}(s_{\ t,}\ a_t) = Q_{\ k}(s_{\ t,}\ a_t)+a(\ r_{\ t}+\ \Upsilon\ \max Q_{\ k}(s_{\ t,}\ a)-Q_{\ k}(s_{\ t,}\ a_t))$$

We define the forward probability, α_k(t) t), as the joint probability of observing the first t vectorsv_t, T = 1. . . t while in state k at time t. Another way to state this would be that (t) is the probability of observing while; in addition, at time t the state is k.

$$a_{\ k(t)}=p\ (v_{1,}\ v_{2,}\ \dots\ \dots\ \dots,\ v_{t,}\ s_t=k\ |\ \Lambda)$$

This probability can be evaluated by the following recursive formula:

$$a_{\ k}(1)=\pi_{\ k}\, b_{\ k}(v_1),\quad 1 \leq\ k \leq N$$

However, when the sequences of observations (the length of the episodes) become larger,
the probabilistic values in the forward algorithm get increasingly small and, after multiple iterations, the values tend to zero. For that reason, αk (t) are scaled during the iterations of the algorithm to avoid underflow problems. The scaling coefficients are used to keep the probability values in

the dynamic range of the machine. So, the coefficient ct is defined as follows

$$c_t = \frac{1}{\sum_{k=1}^{N} \alpha_k\ (t)}$$

Using, the scaled value of αk (t) would be

Given: T (a test suite), PT (a set of permutations of T), and f (a Function that maps PT onto a real number). Problem: Find T´ ϵPT such that

$$(\forall T'')(T''\epsilon PT)(T'' \neq T')[f(T') \geq f(T'')]$$

IV. PROPOSED SYSTEM

The proposed prioritization STRATEGY is BUILT up by JOINING Reinforcement Learning (RL) and Hidden Markov Model (HMM) concepts to efficiently and rapidly prioritize test cases. The main reasons for choosing Reinforcement Learning are its strong statistical background, its proven capacity in handling a extensive range of data, and its capacity to re-estimate the Markov model efficiently. Utilizing RL, we are able to estimate an appropriate HMM and then use it to compute each test case's forward probability, that is, the likelihood of executing a specific test case based upon the SUT's gathered HMM.

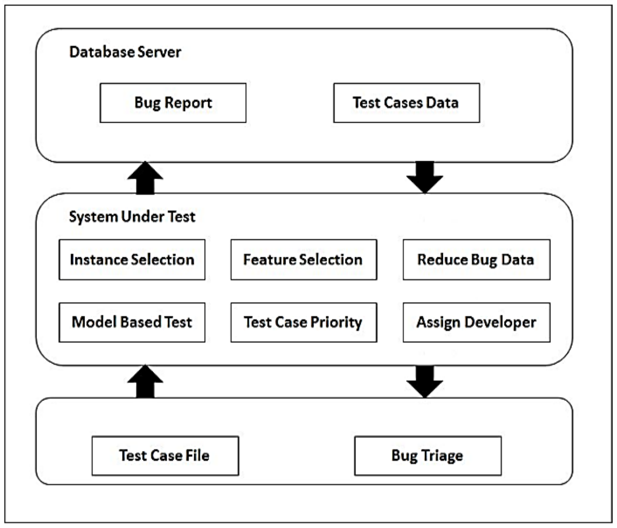

Figure 1: Proposed System Architecture

Advantages of Proposed System

1. They calculate the fault detection rate of every strategy and discovered their proposed methods are able to improve the rate.
2. It delineates that considering GUI states and activities plays a crucial role in improving the fault detection rate.
3. We can reduce the time required to execute test cases and improve the likelihood of consume testing time more beneficially in the case of an unexpected termination of regression testing activities.

V. RESULTS AND DISCUSSION

In this section the results of test case generation for existing (random) method & proposed (prioritization) method are discussed. The following table shows the performance of the existing & proposed system in terms of percentage of test case generation.

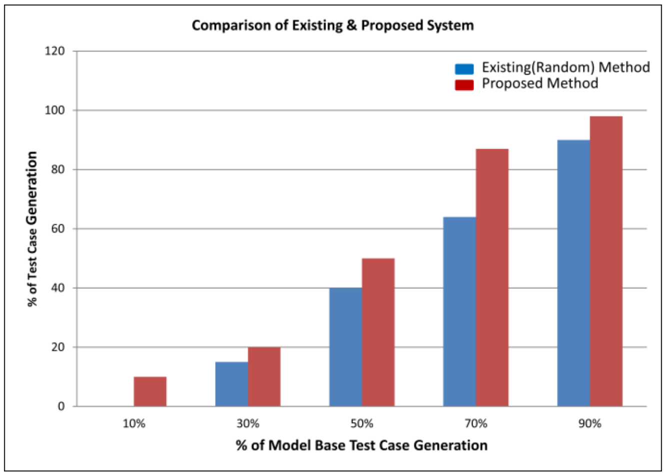

Figure 2: Comparison of Existing & Proposed Method

Result Table:
Table.1: Comparison of Existing & Proposed Method

Module Base Test Case Generation	Random Method Performance (%)	Prioritization Method Performance (%)
10%	0	10
30%	15	20
50%	40	50
70%	64	87
90%	90	98

Comparison of existing and proposed system:

Existing System
Existing system defines the problem of fault detection rate by a measure of how fast a test suite is able to diagnose faults

during testing with aspect of regression testing. Assignment of appropriate developer to solve fault detected in testing.

Proposed System

a) **Techniques**

In existing system random technique is used for test cases and in proposed system we are using prioritization technique for test cases.

b) **Order**

In existing system test case view as Q-values with order of descending sort technique, in proposed system we are using as Q- values with first in first out techniques.

Test case Solving

In existing system using only part of generation test case and schedule test case with help of Q-value and in proposed system bug triage technique for solving test cases.

VI. CONCLUSION

In this paper, we propose a prioritization for Test Case. This paper refers shopping application for generation of test cases using technique of prioritization. We present an approach to initialize an appropriate GUI based on a Q-learning algorithm. Then we use the estimated model to compute the likelihood (forward probability) of the generated test case of forward probabilities. In addition, we propose another technique which uses the summation of each test case's Q-value in order. Thus, test cases with higher amount of accumulated Q-value get higher first in and first out order. First, we intend to perform additional studies on more applications such as Web-based. Second, in this study we only consider GUI applications. We want to evaluate this method further; we are working on presenting a generic approach to generate an RL-based weighted model for every type of application. Such techniques can be utilized to compute the reward function and Q-values in non-GUI-based applications. Third, we need the best sequence of GUI states contributing to the most appropriate prioritized test suite.

REFERENCES

[1]. Gupta S, Raperia H, Kapur E, Singh H, Kumar A. "*A Novel Approach for Test Case Prioritization",* International Journal of Computer Science, Engineering and Applications. Vol.2, Issue.3, pp.48-53, 2012.

[2]. Kumar A, Gupta S, Reparia H, Singh H., "*An approach for test case prioritization based upon varying requirements",* International Journal of Computer Science, Engineering and Applications. Vol.2, Issue.3, pp.87-99, 2012.

[3]. P. R. Srivastava, "*Test case prioritization",* Journal of Theoretical and Applied Information Technology, vol.4, Issue.3, pp.178-181, 2008.

[4]. Chandraprakash Patidar, "*A Report on Latest Software Testing Techniques and Tools*", International Journal of Scientific Research in Computer Science and Engineering, Vol.1, Issue.4, pp.50-52, 2013.

[5]. Suresh Jat, Pradeep Sharma, "*Analysis of Different Software Testing Techniques*", International Journal of Scientific Research in Computer Science and Engineering, Vol.5, Issue.2, pp.77-80, 2017.

[6]. A. G. Malishevsky, J. R.Ruthruff, G. Rothermel, and S. Elbaum, "*Cost-cognizant test case prioritization*", Department of Computer Science and Engineering, University of Nebraska-Lincoln, Techical Report, 2006

[7]. I. Sharma, J. Kaur, M. Sahni, "*A Test Case Prioritization Approach in Regression Testing*", International Journal of Computer Science and Mobile Computing, Vol.3 Issue.7, pp.607-614, 2014.

[8]. A. Shrivastava, S. Rajawat, "*An Implementation of Hybrid Genetic Algorithm for Clustering based Data for Web Recommendation System*", International Journal of Computer Sciences and Engineering, Vol.2, Issue.4, pp.6-11, 2014.

[9]. C. Kaner, "*What is a good test case*". Star East, 16, 2003

[10]. G. Rothermel, R. H. Untch, , C. Chu, M. J. Harrold, "*Test case prioritization: An empirical study*", IEEE International Conference on Software Maintenance, pp.179-188, 1999.

[11]. J. M. Kim, A. Porter, G. Rothermel, "*An empirical study of regression test application frequency*", Software Testing, Verification and Reliability, vol.15, Issue.4, pp.257-279, 2005.

[12]. G. Rothermel, S. Elbaum, , A. G. Malishevsky, P. Kallakuri, X.Qiu, "*On test suite composition and cost-effective regression testing*", ACM Transactions on Software Engineering and Methodology (TOSEM), vol.13, Issue.3, pp.277-331, 2004.

[13]. A. Srivastava & J. Thiagarajan, "*Effectively prioritizing tests in development environment*", ACM SIGSOFT Software Engineering Notes , vol. 27, No. 4, pp: 97-106, 2002 .

[14]. H. K. Leung and L. White, "*Insights into regression testing [software testing]*", Software, Maintenance, 1989., Proceedings., Conference on, IEEE, pp. 60-69, 1989.

[15]. G. Rothermel & M. J. Harrold, "*Analyzing regression test selection techniques*" , Software Engineering, IEEE Transactions on, vol: 22(8), pp: 529-551, 1996.

[16]. S. Elbaum, D. Gable & G. Rothermel, "*Understanding and measuring the sources of variation in the prioritization of regression test suites*", METRICS 2001-Proceedings of Seventh IEEE International on Software Metrics Symposium, pp.169-179, 2001.

[17]. G. Rothermel, R. H. Untch, C. Chu & M. J. Harrold, "*Prioritizing test cases for regression testing*", Software Engineering, IEEE Transactions on, vol.27, Issue.10, pp.929-948, 2001.