Volume 4, Issue 1, January 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Requirement Based Test Case Prioritization in Service-Oriented Web Application Using TOPSIS Framework

Parminder Kaur
Department of Computer Science and Engg.
Lovely Professional University, Phagwara, India

Abstract— As we all aware that in today's competitive world the demand of software engineering is increasing day by day in every field. In software engineering, we have to build the software practically. Software testing is basically an approach which is used to ensure whether the software system does what it is intended to do. Testing phase almost cost more than 50% of the total budget. Complete testing of any software system or even a single program is impossible, so a subset of all test cases is designed which has the highest probability of detecting most of the errors. Regression testing is one of the types of testing which is performed after the modification has been done on the software system in order to check the impact of the modification on other requirements. In Regression testing, if we perform the entire test suite again, then it is not worth doing. Test case prioritization is one of the approaches used to prioritize the test case based on some of the important factors i.e. Requirement based; Fault dependency; History based; Coverage based and Cost based. There are a number of simpler techniques by which we can prioritize the test cases like 1-10 ranking and MoSCoW but for numerous requirements these techniques requires numerous repeats. The objective of this paper is to provide a theoretical study of Test Case Prioritization, Service Oriented Web Applications and TOPSIS framework its advantages, limitations and guidelines to avoid those limitations and how to select test case among the test cases having the same priority.

Keywords— Test Case Prioritization (TCP), Technique of Ordered Preference by Similarity to Ideal Solution (TOPSIS), Service-Oriented Web Application, Software Development Life-Cycle (SDLC).

I INTRODUCTION

Software engineering is a systematic and disciplined approach with the objective of solving client's problem by developing high quality systems within time and budget constraints. If we have to write a small program then we can write it without using any of the software engineering principles but in case of large and complex software we need these so that our product is of better quality, completed in fixed time duration and within budget. There are several stages which must be fulfilled by the software development models. The various stages are Requirement Specification, Software Design, Implementation and Integration, Testing (or Validation and Verification), Deployment (or Installation) and Maintenance. All these steps are known as the SDLC approach. All these steps have their own importance in the development cycle. Testing is one of the important phases of the SDLC approach. In this, we evaluate a system or its components in order to find out whether they satisfy the specified requirements or not. Testing performs the comparison of actual result with that of the expected result.

Every coin has two faces. The testing phase will cost more than half of the total budget. But if testing is not performed on the system and later on error occurs, then it costs even more. Another drawback is that tester may not be able to know which test case to perform first. E.g. suppose a system in which some inputs are taken, after performing functions outputs are generated. If we generate test cases for our system and the test case which found the fault in the output, runs first (i.e. t_1) but the major cause of that fault is because of the input for which the test case is not yet performed so the t_1 test case is of no use. To overcome this problem we use the TCP technique in which we assigned priority to each test cases based on some criteria. The various criteria on which we prioritize the test cases are on the basis of code coverage, fault detection, requirements etc. [1].

II. Test Case Prioritization

Now days, Test Case Prioritization (TCP) approach is increasing popularly. TCP involves scheduling of the test cases in such a way that it will improve the performance of regression testing. Regression testing is one in which we perform the test cases after making modification in the system. So, at that time it is not possible to run all the test cases, we must prioritize the test cases in order to reduce the cost and the time [2]. Regression testing is applied in order to ensure the validity of the system [3].

Performance goals of TCP are-

- Fault detection rate[4]
- Rate of code coverage[5]
- Increase reliability of the system with respect to confidence

Problem statement

Given: T, a test suite; PT, the set of permutations of T; f, a function from PT to the real numbers.

Problem: Find T' belongs to PT such that ($\forall$ T") (T" $\in$ PT) (T" $\neq$ T') [f (T') $\geq$ f (T")].

Here, PT represents the set of all possible prioritizations (orderings) of T and f is a function that, applied to any such ordering, yields an award value for that ordering [6].

Techniques of Test Case Prioritization

There are various different types of techniques on the basis of which we can prioritize the test cases. The technique of test case prioritization can be classified into following four categories and these categories are known as "4C" classification-

- Customer Requirement Based Techniques
- Coverage Based Techniques
- Cost Effective Based Techniques
- Chronographic History Based Techniques

Customer Requirement Based Techniques

Customer Requirement based techniques are the methods in which we prioritize the test cases based on the requirements. There are various frameworks under the requirement based technique by which we set the priority of our requirements-

- 1-10 ranking of each requirement [7]
- Cost of delay
- TOPSIS

Coverage Based Techniques

Coverage based techniques are the methods used for the prioritization of the test cases based on coverage parameter, such as requirement coverage, decision coverage and statement coverage [5]. This testing is a form of glass box testing because it resembles to the code directly. The following lists a process of coverage based techniques-

- Finding that area not exercised/test by a set of test cases.
- Create additional test cases to increase coverage.
- Determine quantitative measure of code coverage.
- Identifying redundant test cases.

Cost Effective Based Techniques

Cost Effective based techniques basically used the methods for test cases prioritization are the cost of analysis and cost of prioritization. Alexey proposed four code coverage heuristic techniques-

- Total function coverage prioritization
- Additional function coverage prioritization
- Total function difference based prioritization
- Additional function difference-based prioritization

Chronographic History-Based Techniques

Chronographic history-based techniques are the methods to prioritize test cases based on test execution history [1]. In this approach, we require the prior performance of each test case in order to increase or decrease the likelihood of test case. It is often most useful in case of Black Box testing where the code information is not available [8].

Importance of Test Case Prioritization

First of all we must be aware of why testing is important. Although, no doubt, testing software costs even more than half of the total budget. But by not performing the testing and later on when error occurs then the cost increases many folds. Not only does it affect the cost but the quality and the customer satisfaction along with the goodwill of the company. So, testing is must for any type of software either small or complex.

Now, as we are aware of the importance of testing, let us consider the case of non-prioritization in which we doesn't set any priority among the test cases. Suppose we have to test any application which has 20 screens on an average of 4 menu buttons on each screen, 3 options on each menu, 10 fields on each screen, 2 types of input per field and around 100 possible input values. Then the total numbers of test cases are-

20*4*3*10*2*100 = 480,000 tests (approx.)

Now let us consider the time required to run these-

Test length = 1 sec then test duration = 17.7 days

Test length = 10 sec then test duration = 34 weeks

Test length = 1 min then test duration = 4 years

Test length = 10 min then test duration = 40 years

This is only an example of very small application and in case there are large and complex one, then we can't even think how many years are required to test a whole application. So we use test case prioritization

- To minimise the test cases.
- To run those test cases that found faults earlier.
- To minimise the time and cost to run.

III. Service-Oriented Web Applications

In recent days, the use of web application is increasing widely. Basically, Web Applications are the applications which are accessing on the web. They are the computer programs which allowed the user to send or retrieve data over the

internet. Service-Oriented Web Application is a type of web application. In the Service-Oriented Web Applications the work is performed at the server side. e.g. Emails, Shopping Cart etc. Service-Oriented Web Applications are the applications where we have numerous requirements and if we prioritize these by using simpler 1-10 ranking or MoSCoW techniques than it requires more effort, time as well as money [9].

Architecture of Service Oriented Web Applications

Service Provider – Service Provider provides the web services and publishes its interface. Provider must decide about security, accessibility, cost and other factors.

Service Broker- Service Broker also known as Service Registry. It is responsible to make available the interface to the service requester.

Service Requester- Service Requester is a client who locates the entries in the broker registry and then binds to the service provider to use the service.

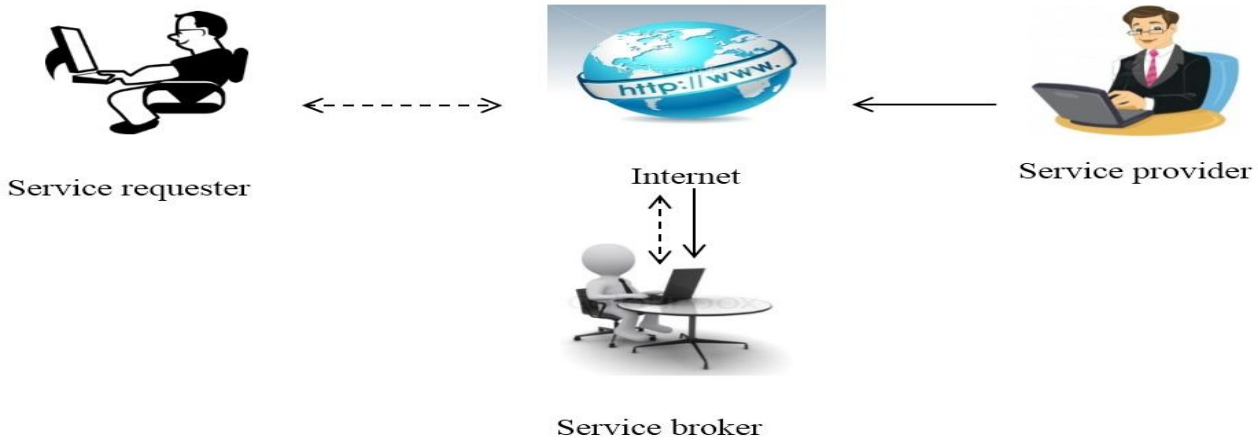

Fig.1 Architecture of Service-Oriented Web Applications

Benefits and Limitations of Service-Oriented Web Applications

Service-oriented web applications are those which provide services to its user or client. These are run on the servers. The benefits and drawbacks of these are-

Table I. Benefits and Limitations of Service-Oriented Web Applications

Benefits	**Limitations**
Same application can be used by different browsers	Have to support different browsers and its versions
Web application is always up to date. No need to download, install and update.	Web applications needs Internet which is not 100% available.
Reduces business costs.	Can take longer to develop as they are more complex.
Direct access to latest information.	Security risks.

IV. TOPSIS Framework

The Technique of Ordered Preference by Similarity to Ideal Solution (TOPSIS) is a decision analysis framework. In this framework, a set of alternatives is evaluated on the basis of a set of criteria. The alternatives may be scored on an absolute or relative scale. In this framework each criteria have a direction of preference. In this the alternatives are ranks so that the vector distance from the ideal solution (S') is minimized and from the non-ideal solution (S*) is maximized.

Characteristics of TOPSIS

- **Attaching weights to different criteria**
  TOPSIS allow for attaching weights to different criteria against which the set of requirements (alternatives) would be prioritized.
- **Absolute or relative scale**
  In TOPSIS alternatives may be scored on an absolute scale (e.g. dollars, effort-hours) or a relative scale (e.g.1-9 Likert scale or Fibonacci scale)
- **Ease of use**
  TOPSIS have a low learning curve so that it can be adoptable widely.

- **Low effort and time overhead**
  TOPSIS requires less time to prioritize the test cases.
- **Scalability**
  TOPSIS can handle large number of requirements without too much effort and time overhead.
- **Tailorability**
  TOPSIS framework is tailor-able meant that it is easy to add/remove existing criteria for prioritization.
- **Ease of Realization**
  TOPSIS not only prioritize on the basis of business value alone it also consider technical feasibility, cost, schedule, effort etc. [9].

Advantages of TOPSIS

When we are developing a Service-Oriented Web Application than there are a number of requirements and we have to prioritize among those in order to deliver a highly valuable system.

- An earlier approach of prioritizing the requirements on the basis of 1-10 ranking or MoSCoW leads to numerous repeats, TOPSIS reduces the repeats by conducting various requirements at once.
- Focuses on the most valuable items first in the various engineering activities.
- Time savings and efficient utilization of resources towards the most valuable requirements.
- Help to compare new requirements with the existing ones to better judge the overall system [10].

TOPSIS Excel™ Template

The initial version of TOPSIS was developed as an Excel™ Workbook. Fig. 2 shows the snapshot of the template. The criteria which we choose to prioritize the set of requirements are captured in topmost row. Each criterion has a relative or absolute score and a direction of preference indicated by '+/- 'symbol. Each requirement is placed under the 'List of Requirements'. In this fig. the requirements are numbered from 1-9. The dependencies/prerequisites are specified by giving each requirement an auto-generated ID. E.g. let's suppose Effort is a criterion for comparing set of requirements. If we want to rank requirement with 'low effort' then effort would be depicted as '–'symbol. If however, we wish to rank requirement with 'high effort' then effort would be depicted as '+'symbol.

Click to Generate Next Level of Prioritization / Delete Current Level / Root	UPDATE / Prerequisites: Comma separated list of requirement IDs/Serial Number(s)	Serial Number (Autogenerated)	Overall Weighted Ranking (Non Dependency Adjusted)	Project Goals/Expectations/Criteria	Criterion 1	Criterion 2	Criterion 3	Criterion 4	Criterion 5
Categorization (Available only if not at root level)	Check box to perform dependecy adjusted ranking ☐	Select Id Prefix (below)	TOPSIS	(Relative) Goal Weights	1	3	3	6	4
				Direction of Improvement (+/-)	+	-	+	+	+
		R	0-1 Normalized Score	List of Requirements/Test-Cases/Alternatives	For each requirement fill in it's relative re Suggested scale: 1-9 (1 = low satisfactior *(You MAY use numbers greater than 9 if y*				
		R1	0.3051	Requirement 1	1	2	3	8	2
	R3, R4	R2	0.1786	Requirement 2	6	4	9	2	5
		R3	0.1597	Requirement 3	3	5	7	1	7
		R4	0.3566	Requirement 4	8	6	5	7	9

Fig. 2 TOPSIS Excel™ Worksheet [10]

Limitation of TOPSIS

Although, TOPSIS received a positive feedback in performing a Multiple Criteria Decision Analysis there were a few limitations. The following limitations are-

- **Limitation 1 –**
  Ability to hierarchically organise the goals (i.e. hierarchal prioritization)
  How to handle this-
  For this, we first prioritize the test cases of the child level after that combine the result for that of the parent level.
- **Limitation 2-**
  Factoring in pre-requisites
  How to handle this-
  Most of the prioritization methods leave the concept of prerequisites due to its surplus complexities. The absence of these pre-requisites changes the priorities. In TOPSIS we can resolve this by making final score of the requirement less than each of its prerequisites.

- **Limitation 3-**
  Rank reversals (i.e. adding irrelevant requirements/test-cases can lead to change the priorities of the existing items which in turns result in a different prioritization ordering.
  How to handle this-
  Irrelevant requirements/test cases must be filtered out before prioritization [10].

Special case
In case if any of the two test cases have same priority then prioritize them again on the basis of the criteria and the alternative that are of most important. If in case again we get same priority then select on the basis of CPU utilization, memory required.

V. CONCLUSIONS

The scope of TOPSIS is to standardize a value focussed mind-set when prioritizing and selecting requirements for systems implementation with large number of requirements. The advantage of TCP is that they don't discard test cases as in the Test Case Reduction along with improvement in testing performance and decrement in regression costs. But TOPSIS framework also has its own limitations. In this paper we present theoretical study about the TCP and the TOPSIS framework in case of numerous requirements such as Service-Oriented Web Application and highlighted the various limitations and provide mitigation guidance to handle those. Then we also point out the selection basis if two of the test cases having same priority. But there is still lack of practical implementation and tools to improve TOPSIS. So, there is a need of web based tool which can handle most of the limitation of TOPSIS and lead to more decrement in regression testing cost.

ACKNOWLEDGMENT

I would like to express my heartiest gratitude to all the people who poured their efforts in compilation of this work. I would like to thank our Almighty God for giving me strength to pull through this task, my parents for supporting me in every step of life and providing me the resources that helped me in my work. I gratefully thank to my friends who helped me a lot in searching the information.

REFERENCES

[1] Lin, Chu-Ti, Cheng-Ding Chen, Chang-Shi Tsai, and Gregory M. Kapfhammer (2013) *"History-based Test Case Prioritization with Software Version Awareness*." International Conference on Engineering of Complex Computer Systems(2013) 18

[2] Askarunisa, A., A. M. Abirami, K. Arockia Jackulin Punitha, B. Karthik Selvakumar, and R. Arunkumar (2010). *"Sequence-based techniques for black-box test case prioritization for composite service testing."* In Computational Intelligence and Computing Research (ICCIC), 2010 IEEE International Conference on, pp. 1-4 IEEE, 2010.

[3] Zhai, Ke, and W. K. Chan (2010). *"Point-of-Interest aware test case prioritization: Methods and experiments."* In Quality Software (QSIC), 2010 10th International Conference on, pp. 449-456. IEEE, 2010.

[4] Garg, Deepak, and Amitava Datta (2012) *"Test case prioritization due to database changes in web applications."* In Software Testing, Verification and Validation (ICST), 2012 IEEE Fifth International Conference on, pp. 726-730. IEEE, 2012.*"*

[5] Kavitha, R., and N. Sureshkumar (2010). *"Test case prioritization for regression testing based on severity of fault."* International Journal on Computer Science and Engineering 2, no. 5 (2010): 1462-1466.

[6] Kayes, Md Imrul. (2011) *"Test case prioritization for regression testing based on fault dependency."* In Electronics Computer Technology (ICECT), 2011 3rd International Conference on, vol. 5, pp. 48-52. IEEE, 2011.

[7] R.Kavitha and Kumar N. Suresh (2011) "Factors oriented test case prioritization technique in regression testing" In proc. of the 2011 IEEE Fourth.

[8] Roongruangsuwan Siripong, Daengdej Jirapun *"Test Case Prioritization Techniques"* Journal of Theoretical and Applied Information Technology, pp 45-60. (JATIT &LLS © 2005-2010).

[9] Kukreja, Nupul, Barry Boehm, Sheetal Swaroop Payyavula, and Srinivas Padmanabhuni(2012) *"Selecting an appropriate framework for value-based requirements prioritization."* In Requirements Engineering Conference (RE), 2012 20th IEEE International, pp. 303-308. IEEE, 2012.

[10] Kukreja, Nupul, Sheetal Swaroop Payyavula, Barry Boehm, and Srivinas Padmanabhuni (2013) *"Value-based requirements prioritization: usage experiences."*Procedia Computer Science 16 (2013): 806-813.

