TEST CASE PRIORITIZATION USING HYPERLINK RANKING- A GRAPH THEORY BASED APPROACH

Krithika L.B [1], Selvakumar R [2], Anand Mahendran[3]

[1]*Assistant Professor, SITE, VIT University, Vellore, India,* ***krithika.lb@vit.ac.in***
[2]*Senior Professor, SAS ,VIT University,Vellore India,* ***rselvakumar@vit.ac.in***
[3]*Associate Professor,SCSE, VIT University,Vellore India,* ***manand@vit.ac.in***

Abstract

Era of cloud computing where majority of the application is becoming web based enterprise computing. User prefer online web application for easy of use and business continuity [1]. Software companies have come up in huge numbers for developing web based enterprise application. Testing is an integral part of any software company which requires more effort. Enterprise applications are complex and navigation is largely based on hyperlink connecting the web pages. Testing phase mostly associated with time constrain to accomplish the task associated in this phase. Main activity of testing phase is execution of test case to test the application. Exhaustive testing is not possible and release a software system without testing the entire application is risk [2]. This paper demonstrates how graph theory can be used to prioritize the test case execution.

Keywords: *Testing, Graph theory application*

----------***----------

1. INTRODUCTION

Software testing immense challenges is that extensive testing is not feasible. Planned time for test phase would often get crunched into shorter deadline and release date, due to unseen delay in the earlier phase of software development life cycle [3].Testing activities in test phase are dependent on the test scenarios and test case written for each scenario.

Tester lead by team leader executes the test case every time the testing phase commences and tries to finish the predefined number of test case to test the application. Test scenarios and associated test case changes are associated with effort, along with the evolving application under test. Below is the Reference Test Case Specification Template. (IEEE 829-1998)[4][5],

Table1: Partial test case template representing only required attribute of the paper

Test case Id	*Test Scenario*	*Test Case*	*Test Steps*	*Result*

Regression testing is a major part of testing phase. Regression testing is to ensure that a change introduced into the software because of new addition or bug fix dose not impact the existing functionality of the system. Build regression and final regression are the two variants of regression testing.

Build regression is a testing that happens without major change and the system is still expecting development. Final regression is the regression testing that happens before the system goes live after many changes to the system.

Graph theory is an area of mathematics that deals with entities (called nodes) and the connections (called links) between the nodes [6][7].Test case prioritization can help to take strategic decision on sequence order [8] but these are static in nature and require revisit every build. In this paper, graph theory is used to model the new approach '*Test case prioritization using Hyperlink ranking'*. This model is more dynamic and accommodates changes dynamically.

2. PROBLEM

Building regression happened after ever new addition to the system. In other words, new features added to the old version between two milestones in a project are carried out to ensure that the system works as expected. The questions that challenges the research at crunching deadline are as below [9],

(a) Do we require executing the regression test suite every build?

The answer is quiet simple, yes

(b) Is there a way to optimize the effort?

We can use stakeholder input to precisely cut down effort

Irrespective to changes happening in the system the effort required for build regression either remain constant without change or effort grows when there is change happening in the application. Using the generic way of testing the web application effort remains same for every version regression testing or may increase. Conventional procedure for testing the application is given in four steps.

2.1 Four Steps Process:

1. Read FRD(Functional Requirement Document) and DD(Design Document)
2. Prepare the Test scenarios
3. Write details test case for each scenarios
4. Test the application based on the detail test case.

Subsequent release with change in the application under test

- Follow Step 1 to Step 4

Subsequent release without changes in the application

- Follow Step 4

Figure 1 shows the flowchart of the above four steps. Any software development faces the problem of deviation in effort estimation in all phase of its life cycle.

Our focus is to reduce the effort required in subsequent regression for web application.

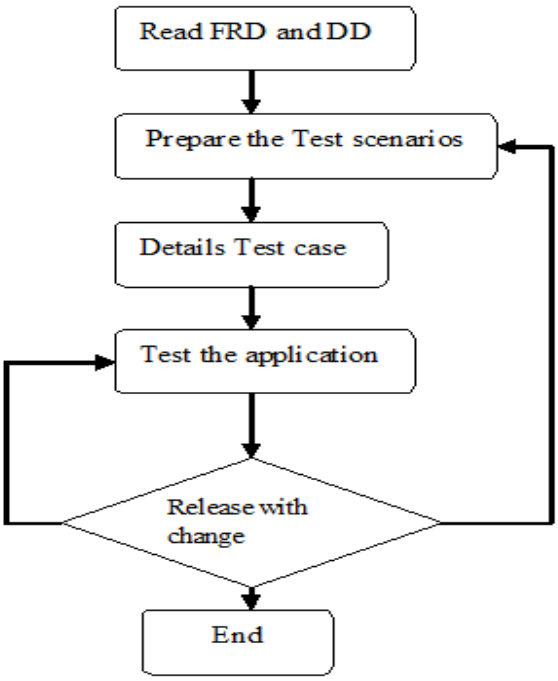

Fig 1: Conventional procedure for testing the application is given by flowchart

3. METHODOLOGY

Application we test can be represented as a graph [10]. Each page in the application are represented as node, pages are connected by hyperlink in each page. The connection between the pages represents the link. The frequencies of usage between the edges are the weight age. Every web application is made up of screen and actions in the screen. The screens are connected to other screen via an action or a hyper link. The user using the application navigates through the application by click the required hyper link to navigate between pages.

We record each user session navigation path for entire usage time. We keep tracking and prepare a repository of user navigation on the entire web application. Using the repository we have the Link map of entire application. Each link is given weight age using simple usage metric.

When a user uses the link between the nodes to navigate we increment the count by one. Every access of the link is increment and stored in the repository. Link graph of the application under test and weight age is based on user interaction data.

4. SOLUTION

In this work, a web application that has 11 pages is considered. These pages are connected bidirectional as per the application flow. In Figure 2, the flow starts from the home page and depending on the user input navigation is chosen across the page and produces the desired result.

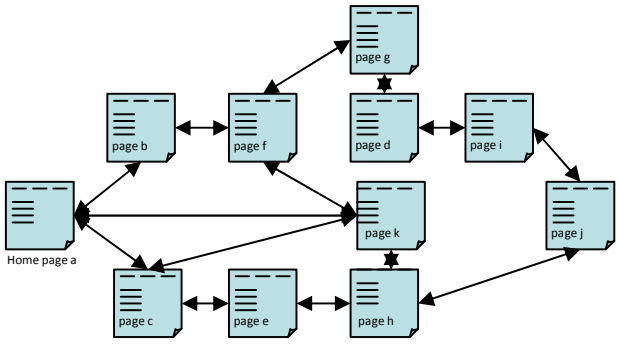

Fig2: Page map of application under test

Application mapped is represented as a graph with link weight. This graph helps us statically traverse the application for experiment. Application graph and Link weight age representation is given as follows in Figure 3,

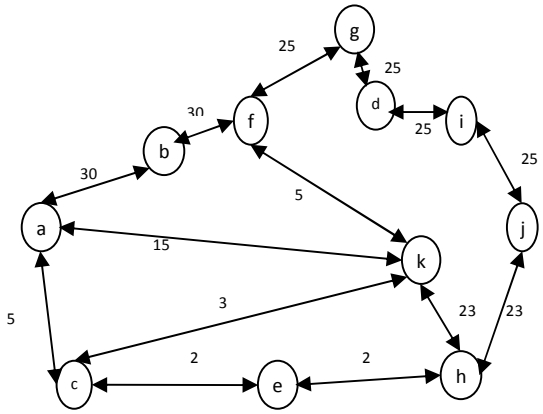

Fig 3: Application graph and Link weight age

4.1 Test case

The sequential step on how to use the application, what input to be given and what result is expected. The deviation in the result as prescribed in the test case is considered to be note for forte action.

Table 2: Repository of Regression suite with link weight

Test case Id	*Test Scenario*	*Test Case*	*weight age*	*Test Steps*	*Result*
A3s1	A_3S= {a},{ab},{ac}	*a*	*35*	…	*P/F*
A3s2	A_3S= {a},{ab},{ac}	*ab*	*30*	…	*P/F*
A3s3	A_3S= {a},{ab},{ac}	*ac*	*5*	…	*P/F*
B2s2	B_2S= {b},{ba},{bf}	*b*	*60*	…	*P/F*
B2s3	B_2S= {b},{ba},{bf}	*ba*	*30*	…	*P/F*
B2s4	B_2S= {b},{ba},{bf}	*bf*	*30*	…	*P/F*
C3s1	C_3S= {c},{ca},{ck},{ce}	*c*	*10*	…	*P/F*
C3s2	C_3S= {c},{ca},{ck},{ce}	*ca*	*5*	…	*P/F*
C3s3	C_3S= {c},{ca},{ck},{ce}	*ck*	*3*	…	*P/F*
C3s4	C_3S= {c},{ca},{ck},{ce}	*ce*	*2*	…	*P/F*
…	…	…		…	…
J2s3	J_2S= {j},{ji},{jh}	*jh*	*23*	…	*P/F*

4.2 Ranking to Links

(a)Procedure

[1] A daemon keeps listing to session at the start of testing
[2] Page link, parent page, source page, to page and the click through is recorded
[3] Count is incrementally store after successful session close.
[4] Data store is a repository of historic data from start of the project to completion.

(b)Pseudo-algorithm

Start: session Start
Capture LinkClickPage && session
usage count ++ &&
session <> same Session
Repository DataStore
End: commit data to store

5. RESULTS

Using our approach we were able to come out with the below scenario and test case associate with priority.

Table 3: Result

Session scenario	**Weight age**	**Priority**
a,b,f,g,d,i,j	160	1
a,b,f,k,h,j	…	…
a,b,j,k,h,j	…	…
a,k,h,j	61	4
a,c,e,h,j	…	…
a,c,k,h,j	54	6
…	…	…

CONCLUSIONS AND FUTURE WORK

Executing build regression is important but can be replaced with thinner version of regression suite that is prioritized based on end user usage. Using the usage metric we were able to effectively reduce the number of test case and also control bug leakage. Comparing to the full regression pack execution for very build regression, our method requires lesser effort, quick in identifying issues in important feature and functionality.

Some very important scenarios are not frequently accessed. As a result, the link weight age for those links would not be recorded or not generated. We are currently employing test leading assistance to identify those scenarios and manually give weight age. In the future work we would like to have a cumulative weight mechanism to optimize and increase efficiency.

REFERENCES

[1] Miller, Michael. Cloud computing: Web-based applications that change the way you work and collaborate online Que publishing, 2008

[2] Myers, Glenford J., Corey Sandler, and Tom Badgett. The art of software testing John Wiley & Sons, 2011

[3] Bresnahan, Timothy, Shane Greenstein, and Rebecca Henderson. "Organizational Diseconomies of Scope and Creative Destruction"

[4] Test Case Specification Template. (IEEE 829-1998)

[5] IEEE Standards Association, Software Engineering standards, vol. 3 of Std. 1061: Standard for Software Quality Methodology, IEEE, 1999 ed., 1999.

[6]Chartrand, Gary (1985), Introductory Graph Theory, Dover, ISBN 0-486-24775-9.

[7]Shirinivas, S. G., S. Vetrivel, and N. M. Elango "Applications of graph theory in computer science an overview" International Journal of Engineering Science and Technology 2.9 (2010): 4610-4621.

[8] Rothermel, Gregg, et al. "Test case prioritization: An empirical study." Software Maintenance, 1999.(ICSM'99) Proceedings. IEEE International Conference on IEEE, 1999

[9] Rothermel, Gregg, and Mary Jean Harrold. "A safe, efficient regression test selection technique." ACM Transactions on Software Engineering and Methodology (TOSEM) 6.2 (1997): 173-210.

[10] Chen, Zheng, et al. "Building a web thesaurus from web link structure." Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval ACM, 2003