INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY
WINGS TO YOUR THOUGHTS.....

Transaction Based Prioritization Strategies for Email Applications

V.Rama Krishna[1], M.R.Narasinga Rao[2], Sk.Sunaina Sulthana[3], T.Geethika[4]

[1]Associate Professor, [2]Professor, [3, 4] Research Scholars
[1,2,3,4] K.L.University
Vaddeswaram, Guntur, AP, INDIA-522502
[1]v.ramakrishna@gmail.com,[2]m.narasingarao@gmail.com,[3]sk.sunaina@hotmail.com, [4]geethika.thatiparthi@hotmail.com

Abstract*: Web Applications change the state based on incoming events. The various events pose a challenge to test the functioning of Email applications because there are a large number of possible event sequences that users can invoke through a user interface. Many models have been developed for testing the Email Applications, but they are not satisfying the needs of the end-users. So, an attempt is made to develop a Single model that is generic enough to test the Email applications. Different prioritization criteria are given in the literature for testing the web application test cases of a test suite. In our study we propose a criterion based on the transactions carried out in the Email Applications. The new criterion is applied on four Email Providing Applications and its fault detection effectiveness is ascertained using APFD metric. Transaction based prioritization provides us with the Email Application which detects faults in the test suite quickly and effectively.*

Keywords: *Email-Application, Single Model, Prioritization criteria, Transactions, Faults, APFD.*

1. INTRODUCTION

In this paper, test cases within a test suite of email applications are prioritized based on APFD metric. Following section covers the components present in our proposed work.

1.1. Email Application

Email is a mailing system, where digital messages are exchanged from an author to one or more recipients. In simple terms, email is an electronic document transmission, provided by many Email applications. Some of the Email applications are Gmail, Yahoomail, Hotmail, Rediffmail etc. In our paper we consider these 4 email applications. These Email applications are the most popular Web-based e-mail services with 300 million-plus users. They are free email services operated by renowned organizations like Microsoft Corporation, Google, etc. They serve millions of users across the world. Include options like audio players, tools, junk filters. They are constantly upgraded and best among leading email service providers. Message storage, size of attachment, security and other services are offered either at free or premium based.

1.2. Prioritization Strategies

Now for these email applications test case prioritization will be done. In previous papers, [1] test case prioritization is done on relevant data basing on some strategies. These strategies prioritize test cases of a test suite. [3-4], [6], [8-10].These strategies are nothing but criteria's. Some of the criteria's are:

- interaction-based
- count-based
- frequency-based

Interaction –based criteria: Test case depends on parameters and values for execution. It has two types:

- 1-way
- 2-way

1-way and 2-way parameter-value interaction coverage techniques select tests to systematically cover PV-interactions between windows. 1-way criterion selects the test which does not appear in previous tests. It covers the new PV, but not the one which is already covered. In the second type, 2-way PV interactions are present. It chooses new PV which is uncovered.

Count-based criteria: It is based on count of number of windows, actions, parameter-values covered.[1][11]. Here there are 3 categories:

- Unique-window coverage
- Action-count based
- PV-count based

INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY
WINGS TO YOUR THOUGHTS.....

- Unique window coverage gives preference to those test cases that cover the most unique windows which are not covered in previous tests. Here it is hypothesized that faults will be exposed when windows are visited and that windows should be visited as-soon-as possible.

- Action-count based criteria prioritizes the number of actions in each test (duplicates are also included).Here preference is given to those that includes maximum number of actions. It can be divided into Action-StoL and Action-LtoS.

- Action-StoL (small to large) gives priority to test cases with smallest number of actions. .
- Action-LtoS (large to small) gives priority to test cases with largest number of actions.

-In Parameter Value count based tests are prioritized by the number of parameters that are set to values in a test case. Duplicates are also included here.

Frequency-based criteria: [1][12]Preference is given to the windows. It observes how many times the window is being accessed in the test cases. The following three criteria's differ in how they view the frequency of presence of a window sequence in a test case and thus produce different prioritized orders.

- MFPS
- APS
- Weighted-Freq

- MFPS (Most Frequently Present Sequence of windows) determines the number of times each sequence appears in the test suite. MFPS gives importance to a particular window sequence.

- APS (All Present Sequence of windows) accommodates all sequences during prioritization. MFPS gives importance only to the frequency of occurrence of a single most frequently present sequence .There is the possibility of losing important information about other frequently present sequences which are not included. So, APS checks frequency of occurrence of all sequences to order the test suite. APS selects test cases based on only one sequence.

- Weighted Sequence of windows (Weighted-freq) assigns each test case a weighted value based on all the window sequence it contains and the importance of the window sequence. Initially, frequency of appearance of each unique sequence of windows in the test suite is identified. And then weighted matrix is built for each unique window sequence. Prioritization criteria improve the rate of fault detection of the test cases over random orderings of tests. All these three criteria's are briefly mentioned here. Apart from these criteria's we have proposed a new criterion namely **"Transaction based criteria".**

2. WORK

A. *Transactions:* Individual operation is said to be transaction. A transaction comprise unit of work performed within a system. It is independent of other transactions. Each transaction succeeds or fails as a complete unit but cannot remain in an intermediate state. [2] All transaction processing which are interactive allows multiple individual operations to be linked together as a single indivisible unit. Transactions checks that all operations are completed without errors. In this paper as we are concentrating on Email Applications each transaction is considered as a "test case".

Transactions can be like:

- Creating an account
- Composing a mail without attaching a file
- Composing a mail with file attachment
- Replying a mail without file attachment
- Replying a mail with file attachment
- Emptying trash/spam/inbox/chats
- Deleting selective emails from trash/ sent/ inbox/ drafts
- Spamming an email and moving to spam folder
- De spamming an email in Spam folder
- Changing some of the settings
- Storing new Email Ids
- Adding to CC,BCC
- Forwarding an inbox email/sent email
- Save as draft
- Moving email to particular folder
- Labeling email
- Adding new email id to chat list

Apart from these there are many such multiple transactions in an Email application. Here we have just given the sample transactions. But, in our work, we consider the first five transactions as test cases. They are:

INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY
WINGS TO YOUR THOUGHTS.....

- Creating an account
- Composing a mail without attaching a file
- Composing a mail with file attachment
- Replying a mail without file attachment
- Replying a mail with file attachment

B. Faults: Mistakes in the code is said to be Fault. A fault is the cause of an error. Software fault lies in software, a hardware fault lies in hardware. In simple terms, error leads to fault. Each application has a default fault matrix which is the representation of a set of faults detected by each test case. Faults in the email applications are seeded manually. Additionally, some naturally occurring faults are also discovered during deployment are also seeded in the application. Faults help in extending functionality and its development. High value for faults means that each test case in the suite is detecting a large number of the faults [1],[3],[7],[13] Low value for faults indicates that each test case detects only a small number of the faults. Here, in our paper faults in each email applications are found out manually. We have found out faults up to the maximum extent. Faults disrupt the user from doing the desired work. It acts like an error. In our work, as we deal with email transactions, in those terms we can say a Fault as the one which interrupts users stepping into the next window. The test cases for our test suites have relatively different fault values for different email applications (some high and some low). We observed that some faults are repeating and some are unique. Some duplicated faults occur with different fault time and some with same fault time. Faults observed for all email applications are given in the below tables.

***C. Time*:** Here faults are found out with respect to time. Time taken for a particular fault to occur is calculated in seconds.[3] Time taken for a specific fault to occur differs from one email application to other. Time is unique for all faults. Time changes for each and every fault in a test case of an email application. As we have already told above that some duplicated faults occur with different fault time and some with same fault time. All these conditions may vary based on following factors:

- Fault type
- Internet speed
- Entry of authenticated data

Following tables 1 to 5 shows the faults occurred and its relative time for the 4 email applications.

Email Provider1: Hotmail

Test Case1: Creating an account

S.NO	FAULTS	TIME(sec)
1	Your name contains characters that are not allowed(F1)	1
2	Your password cant be longer than 16 characters(F2)	1
3	These passwords don't match(F3)	1
4	Password must have atleast 8 characters(F4)	1
5	Please choose a password with mix of lower and upper case letters, numbers, symbols(F5)	1
6	You must provide atleast 2 ways for us to confirm its you(F6)	1
7	Your secret answer must be atleast 5characters long(F7)	1
8	The characters dint match the picture. Please try again.(F8)	3
9	Verify that you have entered the correct information(F9)	1
10	Enter your email address in the format someone@example.com(F10)	1
11	Please enter all the characters you see(F11)	1
12	someone@hotmail.com isn't available(F20)	1

Table 1: Faults occurring during 'Creating Account' in an Email Application

INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY
WINGS TO YOUR THOUGHTS.....

Test Case2: Composing a mail without attaching file

S.NO	FAULTS	TIME (sec)
1	Please enter your email address in the format someone@example.com(F10)	0
2	That password is incorrect. Be sure you're using the password for your account(F12)	3
3	Please type an address in the To box(F13)	1
4	Please verify your account(F14)	2
5	Your message seems to have triggered our junk email filters. Could you?(F15)	2
6	Delivery Status notification(F16)	2

Table 2: Faults occurring during 'Composing Mail' in an Email Application

TestCase3: Composing a mail with file attachment

S.NO	FAULTS	TIME (sec)
1	Please enter your email address in the format someone@example.com(F10)	0
2	That password is incorrect. Be sure you're using the password for your account(F12)	3
3	Please type an address in the To box(F13)	1
4	Please verify your account(F14)	2
5	Your message seems to have triggered our junk email filters. Could you?(F15)	2
6	Delivery Status notification(F16)	2
7	Files with errors will not be uploaded(F17)	3
8	This file exceeds your attachment limit(F18)	1
9	Oops, this email has more than 50files attached(F19)	1

Table 3: Faults occurring during 'Composing Mail with file attachment' in an Email Application

TestCase4: Replying a mail without file attachment

S.NO	FAULTS	TIME (sec)
1	Please enter your email address in the format someone@example.com(F10)	0
2	That password is incorrect. Be sure you're using the password for your account(F12)	3
3	Please verify your account. We have noticed some unusual activity in your account. To help protect you, we have temporarily blocked your account.(F14)	2
4	Delivery Status notification(F16)	2
5	Your message seems to have triggered our junk email filters. Could you?(F15)	2

Table 4: Faults occurring during 'Replying Mail' in an Email Application

TestCase5: Replying a mail with file attachment

S.NO	FAULTS	TIME (sec)
1	Please enter your email address in the format someone@example.com(F10)	0
2	That password is incorrect. Be sure you're using the password for your account(F12)	3
3	Files with errors will not be uploaded(F17)	3
4	Oops, this email has more than 50files attached.(F19)	1
5	The file exceeds your attachment limit(F18)	1
6	Delivery Status notification(F16)	2
7	Your message seems to have junk email filters. (F15)	2
8	Please verify your account. F14)	2

Table 5: Faults occurring during 'Replying Mail with file attachment' in an Email Application

Thus these tables show the faults occurred during those particular transactions in Hotmail. We can see

INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY
WINGS TO YOUR THOUGHTS.....

that there are totally 20 faults. The numbering in the brackets (F1,F2,F3,….) are given for the convenient purpose of our understanding, which represents that it is first fault, second fault and so on till twentieth fault. In this way, faults are found out with time for all the test cases of the remaining email provider applications (Gmail, Yahoo mail, Rediff mail).

D. Fault Matrix: From these faults and time obtained previously from the above tables "Fault Matrix" is constructed.

	T_1	T_2	T_3	T_4	T_5
F1					*
F2					*
F3					*
F4					*
F5					*
F6					*
F7					*
F8					*
F9					*
F10	*	*	*	*	*
F11					*
F12	*	*	*	*	
F13			*	*	
F14	*	*	*	*	
F15	*	*	*	*	
F16	*	*	*	*	
F17	*		*		
F18	*		*		
F19	*		*		
F20					*
Number of Faults	8	5	9	6	12
Time(sec)	14	9	15	10	14

Table 6: Fault Matrix

F1, F2, F3… F20 : Total number of faults

T_1, T_2, T_3, T_4, T_5 : Test cases

*: Fault occurring in a specific test case.

E. Rate of Fault Detection (V_{Ti}): Faults are detected for each test case in the test suite. [3][7][13] Total time taken for each test case is presumed. V_{Ti} is the rate of fault detection. It is calculated using following formula:

V_{Ti} = fault / time Eq.1

As we have found out fault and time initially, now we must calculate V_{Ti}.

$V_{T1} = 8/14 = 0.57$

$V_{T2} = 5/9 = 0.55$

$V_{T3} = 9/15 = 0.60$

$V_{T4} = 6/10 = 0.60$

$V_{T5} = 12/14 = 0.85$

Based upon these V_{Ti} values, test cases will be prioritized in decreasing order:

T_5, T_3, T_4, T_1, T_2

F. APFD: APFD is defined by Rothermel et.al. Rate of fault detection of the prioritization criteria is measured using "Average Percentage of Faults Detected" (APFD) metric [3][4]. APFD evaluates effectiveness of prioritized test suite order. It is calculated by taking the weighted average of the number of faults detected during the run of the test suite. APFD is only possible when faults are available. They are used for evaluation.

$$APFD = 1 - \frac{TF_1 + TF_2 + TF_3 + \ldots + TF_M}{m*n} + \frac{1}{2*n} \quad \text{Eq.2}$$

TF_I : position of the first test case "t" in T

m : number of faults

n : number of test cases

F : fault

T : test suite

Basing on this formula APFD is calculated. But prior to that "TF" should be found out. Now let us find out TF_1 value. In the above fault matrix, we can see that fault F1 is first given by test case T5 (according to priority order t5, t3, t4, t1, t2). Now check the position of T5 in prioritized sequence, which is first. Thus, value of TF1 is 1.Similarly from TF_1 to TF_{11} value is 1. Now, fault F12 is first given by T3 according to prioritized order. So TF_{12}=2. Similar

INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY
WINGS TO YOUR THOUGHTS.....

pattern goes from TF_{13} to TF_{19}. Fault 20 is given by test case T5 so thus $TF_{20}=1$

$TF_1 = 1$ $TF_6 = 1$ $TF_{11} = 1$ $TF_{16} = 2$

$TF_2 = 1$ $TF_7 = 1$ $TF_{12} = 2$ $TF_{17} = 2$

$TF_3 = 1$ $TF_8 = 1$ $TF_{13} = 2$ $TF_{18} = 2$

$TF_4 = 1$ $TF_9 = 1$ $TF_{14} = 2$ $TF_{19} = 2$

$TF_5 = 1$ $TF_{10} = 1$ $TF_{15} = 2$ $TF_{20} = 1$

Upon substituting all these TF values in APFD:

$$APFD = 1 - \frac{1+1+1+1+1+1+1+1+1+1+1+2+2+2+2+2+2+2+2+1}{5*20} + \frac{1}{2*5}$$

$$= 0.82$$

0.82 is the obtained APFD value for prioritized test order.

Now, APFD for non-prioritized test-order is calculated in the same manner.

Non-prioritized order: T_1, T_2, T_3, T_4, T_5

$$APFD = 1 - \frac{5+5+5+5+5+5+5+5+5+1+5+1+3+1+1+1+1+1+1+5}{5*20} + \frac{1}{2*5}$$

$$= 0.44$$

In this case, after calculations we observe that APFD value is high for prioritized sequence (0.82>0.44). Thus, in first email provider application i.e., Hotmail has higher value for prioritized test suite order. In this way, APFD calculations are done for all the email applications. They are summarized in table 7.

3. EXPERIMENTATION AND ANALYSIS

We have tested four email providers by taking five transactions for each application. Total number of faults occurring within those transactions/test cases is discovered by manual seeding. Simultaneously, time is overviewed and in some cases it will be presumed. [3] Time is measured in seconds. After finding out the main components of all transactions i.e., faults and time 'fault matrix' is constructed. From this matrix, fault rates are detected i.e., V_{Ti} using the above said formula. Now, here test case prioritizing will be done. [1] As we have already discussed previously in section 1.2 about the three criteria's mentioned above (interaction-based, count-based and frequency-based). In this work, prioritization criteria will be transactional fault rate detection using APFD metric. 'Average Percentage of Faults Detected' determines the effectiveness of test suite orders either it may be prioritized/non-prioritized. Thus, for measuring this certainty, APFD is chosen due to its effective results. So, in our work we have done this calculation for both the prioritized and non-prioritized test orders.

In the table.7, we can observe that there are four email provider applications namely Hotmail, Gmail, Yahoo mail, Rediff mail. Each one of them has different faults with different timing. Some email applications may have same faults. And some may have different faults. For each email application five test cases or transactions are taken. Initial transaction being 'Creating account' has 8 faults for Hotmail. Whereas for Gmail, Yahoo mail and Rediff mail there are 7, 7, 7 fault respectively.

Similarly for second, third and fourth transactions total number of faults differs. Total time taken significantly varies. For some faults it is 1second and for some it's around 44 seconds (which means that it's taking more time for processing /completing the task or in simple terms we can say that it's consuming user's time making him sit in idle state). The fault which takes less time is relatively much better than other email provider applications. So, from our calculations we found out that Email provider 2(Gmail) has least APFD value i.e., 0.81 which depicts that it detects faults quickly without any time delay. Other email applications also detect faults but when compared to Gmail there is a difference of fraction of seconds. Test suites are incremented 10% in each step and they are executed effectively.

INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY
WINGS TO YOUR THOUGHTS.....

Table 7: APFD CALCULATIONS FOR EMAIL PROVIDING APPLICATIONS

	HOTMAIL					GMAIL					YAHOO MAIL					REDIFF MAIL				
	T1	T2	T3	T4	T5	T1	T2	T3	T4	T5	T1	T2	T3	T4	T5	T1	T2	T3	T4	T5
FAULTS	8	5	9	6	12	7	4	9	6	11	7	6	7	6	14	7	6	7	6	20
TIME	14	9	15	10	14	22	19	25	22	17	44	43	44	43	24	38	37	38	37	26
RATE OF FAULT DETECTION (V_{Ti})	0.57	0.55	0.6	0.6	0.85	0.31	0.21	0.36	0.27	0.64	0.15	0.13	0.15	0.13	0.58	0.18	0.16	0.18	0.16	0.76
PRIORITIZED TEST ORDER	T_5,T_3,T_4,T_1,T_2					T_5,T_3,T_1,T_4,T_2					T_5,T_3,T_1,T_4,T_2					T_5,T_3,T_1,T_4,T_2				
APFD	0.82					0.81					0.84					0.85				
NON-PRIORITIZED TEST ORDER	T_1,T_2,T_3,T_4,T_5					T_1,T_2,T_3,T_4,T_5					T_1,T_2,T_3,T_4,T_5					T_1,T_2,T_3,T_4,T_5				
APFD	0.44					0.42					0.38					0.31				

INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY
WINGS TO YOUR THOUGHTS.....

4. CONCLUSION

In previous papers many criteria's are used for prioritizing the test cases. Interaction-based criteria used unique parameter-values. Count-based criteria considered maximum number of actions and unique windows. And frequency-based criteria considered the most frequently present window, weighted values, etc. These criteria's produced test suite orders for Web Applications. So, as a part of this, we proposed a new criterion 'Transaction' for test case prioritization in order to improve efficiency of testing. Transaction based criteria takes email applications, prioritizes test cases and tests the data for faults. These faults percentage is ascertained by APFD metric. Our testing on email applications produced results which showed that prioritization is important for executing test suites and detects faults effectively. Prioritization improves the efficiency of testing with respect to the four email applications and prioritized test cases are more effective. As a future part we would try to impart work on Commercial Applications like eBay, Amazon by taking multiple transactions together.

REFERENCES

[1] **Renee.C.Bryce, Sreedevi Sampath and Atif.M. Memon, "Developing a Single Model and Test Prioritization Strategies for Event-Driven Software," in *IEEE Trans. On Software Engineering*, vol.X, no.X, Jan. 2011**

[2] **http://en.wikipedia.org/wiki/Databasetransaction**

[3] **Praveen Ranjan Srivastava, "Test Case Prioritization," in *Journal of Theoretical and Applied Information Technology*,2005-2008 JATIT**

[4] **Alexey G.Malishevky, Joseph R.Ruthruff, Gregg Rothermel, Sebastian Elbaum, "Cost-Cognizant Test Case Prioritization," 2006**

[5] **http://www.softwaretestinggenius.com/artical Details?qry=715**

[6] **S.Elbaum, A.Malishevsky and G.Rothermel "Test Case Prioritization: A family of empirical studies," in *IEEE Trans. on Software Engineering*, Feb 2002.**

[7] **J.C.Munson and T.M.Khoshgoftaar, "The Detection of Fault-Prone Programs," *IEEE Trans. On Software Engineering*, vol.18, no. 5,pp. 423-433,1992.**

[8] **R.C.Bryce and A.M.Memon, "Test suite prioritization by interaction coverage," in Proceedings of The Workshop on (DoSTA 2007); ACM SIGSOFT Symposium on the Foundations of Software Engineering. Dubrovnik, Croatia: ACM, Sep. 2007, pp. 1–7.**

[9] **J.A.Jones and M.J.Harrold, "Test-suite reduction and prioritization for modified condition / decision coverage," *Trans. On Software Engineering*, vol. 29, no. 3, pp. 195–209, Mar. 2003.**

[10] **D.Jeffrey and N.Gupta, "Test case prioritization using relevant slices," in *the International Computer Software and Applications Conference*. IEEE Computer Society, Sep. 2006, pp. 411–418**

[11] **J. Lee and X. He, "A methodology for test selection," *Journal of Systems and Software*, vol. 13, no. 3, pp. 177–185, Nov. 1990.**

[12] **M.Grindal, J.Offutt, and S.Andler, "Combination testing strategies: a survey," *Software Testing, Verification, and Reliability*, vol. 15, pp. 167–199, Mar. 2005.**

[13] **D.R.Kuhn, D.R.Wallace, and A. M. Gallo, "Software fault interactions and implications for software testing," *IEEE Trans. on Software Engineering*, vol. 30, no. 6, pp. 418–421, Oct. 2004.**