Journal of Software Engineering

ISSN 1819-4311

Academic Journals Inc.

www.academicjournals.com

Journal of Software Engineering 9 (4): 808-817, 2015
ISSN 1819-4311 / DOI: 10.3923/jse.2015.808.817
© 2015 Academic Journals Inc.

Cost-Benefit Evaluation Model for Automated Testing Based on Test Case Prioritization

Maiqin Cui and Chengyao Wang
School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing, 100083, China

Corresponding Author: Chengyao Wang, School of Computer and Communication Engineering, University of Science and Technology Beijing, No. 30, Xueyuan Road, Haidian, Beijing, 100083, China Tel: +8613681162609

ABSTRACT

Cost-benefit evaluation is an important factor to determine whether a project is suitable for automated testing. This study begins with a discussion on benefits and limitations of automated testing and issues that should be considered before implementation. Based on major influencing factors, this study proposes a novel cost-benefit model for automated testing. In this model, K-means algorithm is proposed to determine the prioritization of automated test cases and the principle of production possibilities frontier is introduced to select the proportion of automated test cases. Experiment results show that this model achieves a more accurate evaluation and the classification and selection of test cases improve the accuracy of this model.

Key words: Automated testing, test case prioritization, cost-benefit, production possibilities frontier, risk aversion

INTRODUCTION

Software testing has become increasingly important with the development of software technology. In software development practice, testing often accounts for as much as 50% of the total development effort, which is pointed out by Budnik *et al.* (2010). Software testing includes more automated testing than before, thus the research on automated testing has increased. However, most studies focus on test framework, test case and test technology, few on cost-benefit. The studies of cost-benefit include the analysis of opportunity cost from economic perspective in Ramler and Wolfmaier (2006), the analysis of benefit from Return on Investment (ROI) perspective in Shi *et al.* (2010), the analysis of benefit from automated testing process perspective in Yu (2003) and a search-based approach for deciding what parts should be tested automatically in Amannejad *et al.* (2014). They analyzed the cost-benefit from different perspectives and provided help for automated testers. But they have common shortcomings: (1) The degree of difficulty of automated test cases is ignored, (2) Some major factors are easy to be overlooked in the modeling process, such as test design, (3) Some restrictions are not considered thoroughly, such as time limits, (4) Details of cost calculation method need be provided and (5) Risk is ignored. Our study proposes a cost-benefit model for automated testing from major influencing factors perspective. This model includes test case classification method based on degree of difficulty and test case selection method within time limit. The purpose of this study is to help small and medium enterprises get a more accurate assessment, so that they can correctly decide whether automated test should be implemented and when to implement.

MATERIALS AND METHODS

With the development of software test technology, automated testing becomes popular among testers. In the meantime, it also brings issues such as test scripts reusability, high maintenance and more cost. Admittedly, automated testing can improve test efficiency under certain conditions, but this does not mean that automated testing can completely replace traditional manual testing.

Benefits and limitations: In Rafi *et al.* (2012), the authors analyze the benefits and limitations of automated testing by literature review and practitioner survey. Combining with practical experience, the benefits and limitations are summarized as follows:

- Benefits are reflected in: (1) Scripts have high reusability, (2) Running time is very short and test cost can be reduced after achieving a certain number, (3) Test coverage is improved, so as to improve the quality of product, (4) Some bugs, which are ignored by manual tester's careless or overworked, can be found, (5) Testers can understand entire project well and find some bugs that are caused by non-standard development, (6) It is convenient and highly efficient to do regression testing and (7) Some special tasks, such as performance testing, only can be completed by using automated testing
- The limitations are reflected in: (1) In the early days, it is requisite to invest for buying tools and training testers, (2) Specialized persons are needed to design test framework, test cases and maintain test script, in other words, the technical requirements for testers are higher than manual testing, (3) It is not suitable for short-term projects, (4) Automated testing and manual testing have own strengths in finding bugs. Since automated testing does not have imagination, many new bugs require manual test to discover, (5) Automated testing cannot completely replace traditional manual testing. It is not worthwhile to use automated testing for some test cases, such as some GUI test cases and (6) Automated testing has great dependence on scripts quality

Consideration before implementation: Automated testing has both benefits and limitations. Not all projects are suitable for automated testing and not all test cases can use automated testing to achieve. Sahaf *et al.* (2014) discusses when to automate software testing and Taipale *et al.* (2011) discusses the trade-off between automated and manual testing. In general, the following aspects should be considered before implementation:

- **Whether project team has ability to carry out automated testing:** That is, whether there are appropriate test tools, whether there are competent testers, whether there is suitable and mature test framework, whether the project has been stable, whether early investment could be provided and so on
- **Project cycle and test running frequency:** It is not cost-effective to use automated testing for short-term project. The return cannot be gotten until the number of test run reaches a certain value
- **The degree of difficulty of automated testing:** Because automated testing has limitations on applicable areas, some cases are not suitable for automated testing. In addition, non-standard development will increase the difficulty of test
- **Maintenance workload:** It is very unwise to implement automated testing before project is stable. The change of function points will increase maintenance workload. In addition, quality of scripts will affect maintenance workload

- **Cost-benefit assessment:** Cost-benefit assessment can help to decide whether automated test should be implemented and when to implement

Automated test case prioritization: There are different degrees of difficulty for the same test case when selecting different test tools. Therefore, appropriate tools should be selected before determining test case prioritization. Three factors should be considered: First, open source or existing tools should be preferred, which help to save the cost of buying tools. Second, the tools that testers are familiar with should be prioritized, which help to save training costs. Third, tools should be selected according to project features. Each tool has own characteristics, for example, Selenium is only available for Web test and QTP is available for both Web test and client application test. But Selenium is open source and Python, Ruby, Java, C# can be used to develop scripts, QTP only supports VB.

After selecting test tools, test cases should be classified according to the degree of difficulty (Song and Zhang, 2006). In this study, K-means clustering algorithm is proposed to determine test case prioritization.

The degree of difficulty of test cases depends on three factors: Scripts amount (SN, unit: rows), modification rate of recorded scripts (MR) and script reuse rate (RU). In order to neutralize the weigh in K-means algorithm, each factor is normalized before forming a three-dimensional vector:

$$s = (n, m, r)$$

where, n represents SN, m represents MR and r represents the reciprocal of RU.

Suppose there are N test cases. The original dataset is $S = \{s_1, s_2,..., s_N\}$, the number of cluster group is k (k<N), the algorithm to determine test case prioritization is as follows:

Step 1: Taking k elements from S randomly as the center of k clusters ($x_1, x_2,..., x_k \in S$)

Step 2: For the remaining elements, like element i, calculating the distance between i and the center of all cluster groups. The element is classified to the closet cluster. The x_j represents the j-th cluster center. The formula is shown in Eq. 1:

$$C^{(i)} := \min_j \left\| s_i - x_j \right\|^2 \quad (1)$$

Step 3: According to results, recalculating the center of k clusters. Taking the j-th cluster as an example, suppose there are p elements in the j-th cluster, the formula is shown in Eq. 2.

$$x_j = \left(\frac{\sum_{i=1}^{p} n_i}{p}, \frac{\sum_{i=1}^{p} m_i}{p}, \frac{\sum_{i=1}^{p} r_i}{p} \right) \quad (2)$$

Step 4: Repeating step 2 and 3 until the results does not change in two consecutive iterations

Step 5: Printing the cluster results. Test cases have higher prioritization, which are included in a cluster that has a smaller center value

Let's use a simple example to illustrate application of K-means in classifying test cases. Suppose a project has 10 cases, the original and normalization dataset is shown in Table 1. The number of cluster group is 3.

Table 1: Original and normalization dataset

	SN		MR	RU		
Cases	Data	n	m	Data	1/data	r
s_1	30	0.60	0.60	1.00	1.00	0.10
s_2	25	0.50	0.35	0.50	2.00	0.20
s_3	10	0.20	0.30	0.60	1.67	0.17
s_4	50	1.00	0.50	0.45	2.22	0.22
s_5	36	0.72	0.70	0.20	5.00	0.50
s_6	21	0.42	0.45	0.10	10.00	1.00
s_7	8	0.16	0.50	0.70	1.43	0.14
s_8	15	0.30	0.60	0.90	1.11	0.11
s_9	42	0.84	0.80	0.85	1.18	0.12
s_{10}	24	0.48	0.65	0.30	3.33	0.33

Table 2: Process and result of clustering

Parameters	Cluster 1	Cluster 2	Cluster 3
R1			
Center	s_1	s_2	s_3
Set	$\{s_1, s_4, s_5, s_8, s_9, s_{10}\}$	$\{s_2, s_6\}$	$\{s_3, s_7\}$
Center	(0.66, 0.64, 0.23)	(0.46, 0.4, 0.6)	(0.23, 0.4, 0.16)
R2			
Set	$\{s_1, s_4, s_5, s_9, s_{10}\}$	$\{s_6\}$	$\{s_2, s_3, s_7, s_8\}$
Center	(0.73, 0.65, 0.25)	s_6	(0.29, 0.44, 0.16)
R3			
Set	$\{s_1, s_4, s_5, s_9\}$	$\{s_6\}$	$\{s_2, s_3, s_7, s_8, s_{10}\}$
Center	(0.79, 0.67, 0.24)	s_6	(0.32, 0.48, 0.18)
R4			
Set	$\{s_1, s_4, s_5, s_9\}$	$\{s_6\}$	$\{s_2, s_3, s_7, s_8, s_{10}\}$

Selecting s_1, s_2, s_3 as the center of 3 clusters, the process is shown in Table 2.

Table 2 shows that the values are smaller in cluster 3 than corresponding values in cluster 1. Namely, test cases prioritization is higher in cluster 3 and these cases should be achieved first. Because the script reuse rate is too low, s_6 is assigned in a separate cluster. Automated testing should be given up for s_6.

Test case selection: In practice, selecting test cases need to consider available time. In this study, the principle of production possibilities frontier in economics is introduced to select automated test cases, this principle has been elaborated in Mankiw (2008). Because running time of automated test is very short, running time is ignored. Suppose the available test time is T, the average time of each test case for manual testing is C_M, the total number of test cases for manual testing is N_M which is the sum of all manual test cases running frequency, the number of cluster group is 4, the average time of each group for automated testing is as follow: Level 1 is ∞ which represents they are only tested manually, level 2 is C_1, level 3 is C_2 and level 4 is C_3, accordingly, the number of test cases is N_1, N_2 and N_3. The actual time T' satisfies the Eq. 3 and 4:

$$T' = C_M N_M + \sum_{i=1}^{3} C_i N_i \quad (3)$$

$$T' \leq T \quad (4)$$

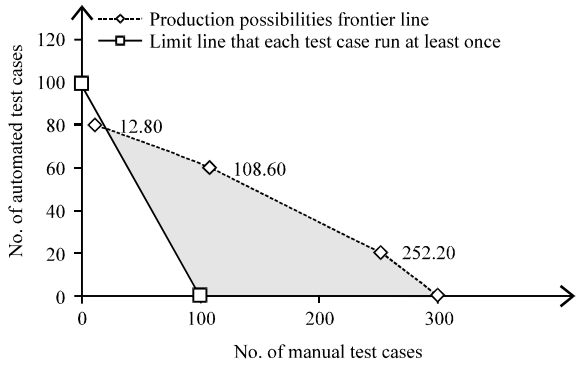

Fig. 1: Production possibilities curve with restrictions

Table 3: Automated test case classification results

Level	Number	Average time	Optional test methods
1	20	∞	Manual
2	20	1.2	Manual or automated
3	40	0.9	Manual or automated
4	20	0.6	Manual or automated

Suppose a project has 100 test cases, the available time is 75 h and the average time of each test case for manual testing is 0.25 h. Automated test case classification results are shown in Table 3.

Automated test cases are achieved according to the priority order. That is, the high priority cases are achieved first. The corresponding production possibility frontier line is the dotted line in Fig. 1. But there are some restrictions in practical applications, for example, all test cases must run at least once. Thus some limit lines should be added in practical applications. In Fig. 1, the straight line is the limit line that each test case must run at least once. Limit lines can be set according to project requirements.

In Fig. 1, the shaded area is the effective area of test cases within the limited time of 75 h. In theory, any value of shadow area can be selected. However, in order to make full use of test time, it is more appropriate to select the value on production possibilities frontier line. Team needs to select an appropriate value based on actual situation. Automated testing can ensure test cases run any times, but because of lack of imagination, the new bug is hard to find. Manual testing may reduce the number of test running, so that some bugs may be missed.

Influencing factors of cost-benefit: The maximum benefit of automated testing is short run time and low cost. Compared to manual testing, the cost-benefit of automated testing is mainly affected by several factors:

- **Test tools:** Automated testing must have tools to support. Some automated testing tools and script development tools require software licenses. The costs can be estimated by purchase cost multiplied by depreciation rate
- **Training:** More professional testers are needed to achieve automated testing. Automated testers need to accept the training of tools and development standardization. The costs can be estimated by training time (unit: person day) multiplied by pay of unit time

- **Test design and cases development:** In the automated test process, tools only play a supporting role, the design of test framework and test cases play a main role. The standard of development directly determines whether automated testing is successful. The costs can be estimated by design time (unit: hour) multiplied by pay of unit time
- **Test running:** Test running is the process of running test cases and analyzing test results. The costs can be estimated by running time (unit: hour per round) multiplied by running frequency multiplied by pay of unit time
- **Cases maintenance:** When a new version is released, the function and interfaces of software may change. Accordingly, test cases need to modify to accommodate new version. The costs can be estimated by the sum of software change coefficient multiplied by the costs of cases design

Model: After selecting automated test cases, this study proposes a simple cost-benefit calculation formula in Eq. 5. The cost-benefit is gotten by the difference between costs of automated testing and that of manual testing. For more intuitive representation cost-benefit, the result is expressed as the negative of the difference between them. If the result is greater than 0, the cost of automated testing is less than that of manual testing, namely the implementation of automated testing is cost-effective. On the contrary, if the result is less than 0, the implementation of automated testing is not worthwhile:

$$CB(n) = -C_t * \alpha - C_s - (C_{ae} - C_{me}) - \left(1 + \sum_{i=1}^{a} \beta_i\right)(C_{ad} - C_{md}) - (C_{ar} - C_{mr}) * n \quad (5)$$

where, CB is cost-benefit of automated testing, n is the number of test running, C_t is purchase cost of tools and licenses, α is tool depreciation rate, C_s is training cost of automated testers, C_{ae} is the cost of test design and framework design for automated testing, C_{me} is the cost of test design for manual testing, C_{ad} is design cost of automated test cases, C_{md} is design cost of manual test cases, a is the number of software change, β_i is software change coefficient, C_{ar} is the cost that automated scripts run once and C_{mr} is the cost of manual test once.

In Eq. 5, the cost is estimated by time multiplied by pay of unit time, automated test case design time is the sum of each level case. The formula of tool depreciation rate is shown in Eq. 6:

$$\alpha = \frac{\text{Test time of project (Unit : month)}}{\text{Durable years} * 12} \quad (6)$$

Theoretically, automated testing is in revenue state when CB (n)>0. However, automated testing has high risk and high failure rate. Non-standard design and development may lead to a great maintenance cost. In practice, in order to better avoid risk, automated testing will not be implemented unless the yield rate γ is greater than a certain value. The γ is determined depending on the actual project, generally take 0.1. The formula of yield rate is shown in Eq. 7:

$$\gamma = \frac{CB(n)}{C_{me} + (1 + \sum_{i=1}^{a} \beta_i) * C_{md} + C_{mr} * n} \quad (7)$$

RESULTS AND DISCUSSION

In this section, a project with practical application is introduced to verify this model and the experiments are designed to answer the following research questions:

- **RQ1:** Can this model evaluate the cost-benefit for automated testing in small and medium projects accurately?
- **RQ2:** What are the advantages of this model, compared with other studies?

Project studied: An advertising company intends to open up micro-blog promotion platform. There are 6 testers in the project team, one of them has experience in automated testing, two have development experience but do not have experience in automated testing and the remaining do not have development experience and mainly do manual testing. Since this project is Web test and testers are more familiar with Selenium, so testers choose Selenium as test tool. The project has 640 test cases and will take 12 months. Testers need to test nine kinds of browsers: The browser of IE7 kernel, Chrome, Firefox, IE8, IE9, IE10, 360, Sogou and Safari and four kinds of OS: Windows XP, Win7, Linux and Mac OS. Test cases need to run 20 times in one round by analyzing the common browsers of each OS. In order to ensure software quality, each version must perform at least four rounds: Smoke testing, system testing, regression testing and α testing. The first version requires 6 months and the first 4 months will be taken into test design, framework design, staff training, waiting for developers to submit code, That is, there are only two months to design and run test cases. The effective working time is about 160 h/person/month.

Experiment: First, the project is estimated using the model in Shi *et al.* (2010). Suppose the average time to design a manual test case takes 0.4 h and the average time for automated testing takes 1 h. The average time to run a manual test case takes 0.05 h, running automated test cases and analyzing test result takes 10 h. The corresponding cost of manual testing and automated testing is shown in Table 4. Smoke testing need to run once and the remaining three rounds need to run 20 times, so the first version needs to run 61 times. The calculation result of ROI is shown in Eq. 8:

$$\text{ROI}(61) = \frac{(0+256+32*61+3*61)-(15+640+10*61+13*61)}{(15+640+10*61+13*61)} = 0.16 \quad (8)$$

Equation 8 shows that the ROI of automated testing is 0.16, that is, automated testing is more effective.

Next, the project is estimated using the model in this study. Suppose the number of cluster group is 4, the classification results using k-means clustering are shown in Table 5. The effective time of 6 people within two months is 1920 h. From the hypothesis in Table 4, the average time of manual test is 0.06 h. In order to ensure that developers have enough time to modify bugs, the development time of automated test cases just has 1 month, that is, 480 h.

Table 4: Costs of manual and automated testing

	Test mode	
Parameters	Manual testing	Automated testing
Hardware and software cost (h)	0	15
Development cost (h)	256	640
Running cost (h)	32	10
Maintenance cost (h)	3	13

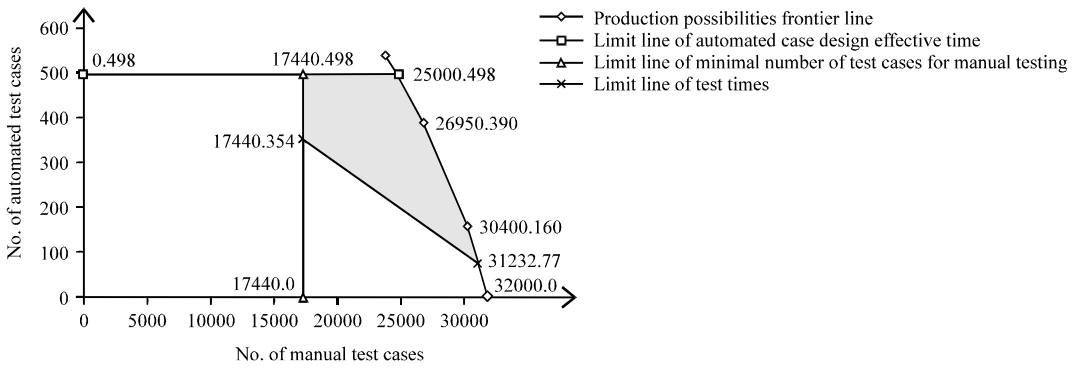

Fig. 2: Production possibility frontier curve with restrictions

Table 5: Test case classification results of present project

Level	Number	Average time (h)
1	100	∞
2	150	1.2
3	230	0.9
4	160	0.6

Table 6: Costs estimation results

Cost items	Cost (*10^4)
C_t	0.20
C_s	1.00
C_{ae}	2.20
C_{me}	0.20
C_{ad}	2.10
C_{md}	0.70
C_{ar}	0.03
C_{mr}	0.12

There must be a round which is tested manually and test cases that cannot be achieved by automated testing must be tested manually, that is, the number of manual test cases is 17,440 at least. At the same time, the task of 61 times must be completed. The production possibility frontier curve with restrictions is shown in Fig. 2.

Figure 2 shows that the optional range is 77-498, here 498 is selected. Suppose the salary of tester who has automated testing experience is 8000 per month, the salary of tester who has development experience is 5000 per month and the salary of manual tester is 4000 per month. The costs are shown in Table 6.

The change frequency of first version is the number of test rounds, corresponding change coefficient are 0.2, 0.1, 0.05 and 0.02. The tool is durable for 3 years, so the tool depreciation rate is 0.17. The calculation result of cost-benefit is shown in Eq. 9 and yield rate is shown in Eq. 10:

$$CB(61) = -0.2\times0.17-1-(1+0.37)\times(2.1-0.7)-(2.2-0.2)-(0.03-0.12)\times61 = 0.448 \quad (9)$$

$$\gamma = \frac{CB(61)}{0.2+1.37\times0.7+0.12\times61} = 0.05 < 0.1 \quad (10)$$

The results of Eq. 9 and 10 show that automated testing is in revenue state, but the yield is less than 0.1. That is, automated testing has relatively large risk in the first version and it is best to use manual testing.

In the second version, another round regression testing is added because of increasing amount of users, the corresponding change coefficient are 0.1, 0.03, 0.02, 0.02, 0.01. The calculation result of cost-benefit is shown in Eq. 11 and yield rate is shown in Eq. 12:

$$CB(81) = -0.2\times0.17-1-(1+0.18)\times(2.1+0.7)-(2.2-0.2)-(0.03-0.12)\times81 = 2.268 \quad (11)$$

$$\gamma = \frac{CB(81)}{0.2+1.18\times0.7+0.12\times81} = 0.21 > 0.1 \quad (12)$$

The results of Eq. 11 and 12 show that automated testing can be used in the second version.

Results and comparison: The experiment results show that automated testing is more effective in both two versions in Shi *et al.* (2010). But for the model in this study, automated testing has relatively large risk in the first version and should be used in the second version.

In practice, testers introduce automated testing in the first version. In the process of implementation, since the change coefficient of script is large, part of the script is abandoned and testers only complete two-thirds of test tasks at six months. By calculating the data as above, if testers use manual testing they can complete four-fifths of test tasks. Besides, since testers will be more familiar with test cases and the bug will reduce in the late, the actual completion will be more than four-fifths. Compared to the first version, the change coefficient is up to 0.6 in the second version. The scripts in the first version are immature and they eventually abandoned. Practice has proved that it is more appropriate to introduce automated testing in the second version, which is consistent with the evaluation results of our model.

It is undeniable that published studies, likely Shi *et al.* (2010), Ramler and Wolfmaier (2006) and so on, provide help for tester to evaluate cost of automated testing. But the evaluation results of these models have a big deviation with actual results. There are several reasons for the deviation. First, automated test cases are not classified. Testers spend too much time on difficult cases and scripts are not reused eventually, but easy cases are not achieved in the last because of lack of time. Furthermore, some major factors are ignored, for instance, the design costs of test framework in Shi *et al.* (2010). Finally, risk of automated testing is ignored and testers may introduce automated testing when the version is not stable. This study compensates for the three defects mentioned above and makes a more accurate evaluation for small and medium projects.

CONCLUSION

This study proposes a cost-benefit evaluation model for automated testing from major influencing factors perspective. The model provides the following ideas: (1) A novel method to determine automated test case prioritization is proposed, which leads tester to achieve test cases from easy to difficult. Automated test cases are selected accurately to ensure the completion of test tasks within limited time, (2) Risk aversion is introduced to avoid risk of automated testing, so that automated testing can be used at right time, (3) In this model, the major influencing factors of automated testing are considered and the calculation method of each factor is elaborated and

(4) This model is easy to understand and assess and cost calculation is simple. In future work, in order to improve estimation accuracy, the study will focus on the estimation method of major factor by using historical data.

ACKNOWLEDGMENT

The authors are grateful to AdSage Company that provide the cases and verify results. At the same time, thank Tianxin Zhao for helping us to correct grammar mistakes.

REFERENCES

Amannejad, Y., V. Garousi, R. Irving and Z. Sahaf, 2014. A search-based approach for cost-effective software test automation decision support and an industrial case study. Proceedings of the IEEE 7th International Conference on Software Testing, Verification and Validation Workshops, March 31-April 4, 2014, Cleveland, OH., USA., pp: 302-311.

Budnik, C.J., W.K. Chan and G.M. Kapfhammer, 2010. Bridging the gap between the theory and practice of software test automation. Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering, Volume 2, May 1-8, 2010, Cape Town, South Africa, pp: 445-446.

Mankiw, N.G., 2008. Principles of Economics. 5th Edn., Cengage Learning, USA., ISBN-13: 9780324589979, Pages: 904.

Rafi, D.M., K.R.K. Moses, K. Petersen and M.V. Mantyla, 2012. Benefits and limitations of automated software testing: Systematic literature review and practitioner survey. Proceedings of the 7th International Workshop on Automation of Software Test, June 2-3, 2012, Zurich, Switzerland, pp: 36-42.

Ramler, R. and K. Wolfmaier, 2006. Economic perspectives in test automation: Balancing automated and manual testing with opportunity cost. Proceedings of the International Workshop on Automation of Software Test, May 20-28, 2006, Shanghai, China, pp: 85-91.

Sahaf, Z., V. Garousi, D. Pfahl, R. Irving and Y. Amannejad, 2014. When to automate software testing? Decision support based on system dynamics: An industrial case study. Proceedings of the International Conference on Software and System Process, May 26-28, 2014, Nanjing, China, pp: 149-158.

Shi, Y.L., Y.Y. Chen, X.S. Luo and Z.G. Jiang, 2010. Cost benefits analysis of automated testing plan. Microcomput. Inform., 6: 218-228.

Song, B. and Z.N. Zhang, 2006. Software automation testing feasibility analysis in system functions testing phrase. Comput. Applic. Software, 22: 31-33.

Taipale, O., J. Kasurinen, K. Karhu and K. Smolander, 2011. Trade-off between automated and manual software testing. Int. J. Syst. Assurance Eng. Manage., 2: 114-125.

Yu, X.S., 2003. Cost benefits analysis of automated testing. Comput. Eng. Applic., 17: 107-109.