Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 79 (2016) 8 – 15

Procedia Computer Science

7th International Conference on Communication, Computing and Virtualization 2016

The Impacts of Test Automation on Software's Cost, Quality and Time to Market

Divya Kumar[a*], K. K. Mishra[b]

[a,b]*Motilal Nehru National Institute of Technology Allahabad, Allahabad,244001, India*

Abstract

In spite of the availability of most proficient quality assurance teams and tools, software testing has always been a time-consuming task. Thus test automation is being profoundly practiced in most of the software industries to leverage the total development time. Although the test automation has its own advantages and disadvantages and it influences various other development phases, the higher management is particularly interested in reckoning its effects on total software's cost, quality and time. In this paper, we have tried to ascertain some of the critical factors related to test automation and cost/return of/from automation. As automation is itself a pricy activity, it requires development effort and significant time, we have attempted to enumerate test automation's impacts on software's cost, time and quality on three different softwares. The results of our experiments clearly show the positive effects of test automation on cost, quality and time to market of the software.

Keywords: Test Automation; Software Development Cost; Quality Factors; Software's Time to Market; Return on Investment.

1. Introduction

Software Testing is the process to bring on the latent defects into identifiable ones. This crucial phase of the software development life cycle uncovers the hidden defects in a software product. Regardless of time-consuming and resource-hungry nature of testing, we can never ignore it. Every newly developed or modified engineering product is required to pass rigorous tests so as to ensure the quality of the developed product[5]. Software Testing utilizes approximately 40%-50% of total resources, 30% of total effort and 50%-60% of the total cost of software development[1–4]. Testing phase, being a major challenge in software development, can be considered as a fair

[*] Corresponding author. Tel.: +91 941 109 5353
E-mail address: divyak@mnnit.ac.in

doi:10.1016/j.procs.2016.03.003

opportunity that can considerably help to improve and optimize software's cost, quality and time to market. This improvement is much desired in the present scenario when software industries are facing tough international competition and trying to shrivel their budgets and schedules [4].

In the interest of this research paper we have classified the Software testing into two basic categories: a) Manual Testing and b) Automated Testing. Since long and now also, we are conducting manual testing of software products; in this type of testing a human tester executes the application and initiates various tests over it by interpreting and analyzing the behavior of the product on various input conditions[2,6]. The human tester later prepares the reports and provides comments on the quality-state of the product by comparing the actual results against the expected results. On the other hand an Automated Testing (AT) refers to the use of some standard software solutions to control the execution of test-cases on the Software Under Test (SUT)[7,8]. This process also involves setting up the preconditions, matching the actual results against the predicted ones and then documenting the observations according to some standard protocol[9,10]. Automated testing requires writing up some special computer programs to find bugs or defects in SUT. It is an excellent approach to replace the laborious and time consuming manual testing. Automated testing has various advantages and it is always suggested for the quality improvements of the application as it provides formal test coverage, avoid human errors and speed up the test execution process[11]. Also, as it speeds up the execution process, it is most effective solution for meeting the strict deadlines.

As a result, today, there are many commercial software tools, that allows fully automation, are available for testing purposes and lot of organizations are engaged in providing the quality assurance/testing services. But, test automation is a very critical process; a lot of factors, like which feature requires automated testing, are needed to be determined before any organization proceeds for the automation testing. Also, test automation requires high primary investments in terms of software feature analysis, scripting, tool procurement and training etc. Thus a precise analysis of the Return on Investment (ROI) from test automation is required before we start[5,12,13].

In this paper we have tried to identify and quantify the ways in which the test automation affects the three critical software dimensions of time, cost and quality. The flow of our paper goes in the following manner: First we have discussed the empirical annotations from the past studies over test automation. Then we have tried to model the effects of test automation on cost, quality and time to market of the software product in the subsequent sections. Afterwards the effects of test automation are calculated on three different software's, through this model.

2. Literature Review

Industrial softwares are released out in versions. For any version there exists many in home local builds of that version at developer's site. Thousands of software developers work together to brought up any single software. In this industrial scenario, developers usually work on the local version of the software on their machine, they modify it repeatedly, and merge their changes with the latest build ready for release. When thousands of developers simultaneously develop nightly (or weekly or monthly builds) it is nearly impossible to test every developer's local modified version build before it is merged to the main build. As the changes are generally iterative test automation is a right answer. The software grows more with more functionality added in each release. Then, why and how much to test the pre-existing functionalities is the difficult to decide.

A right degree of test automation of the pre-existing functionalities is always necessary, cost effective and time saving activity. Ramler et al in[4] discussed the benefits arising from the automated testing. They discussed test automation as one solution to reduce recurring testing costs and proposed a cost based model, to decide on automation strategy. In[14] Berner et al. claimed that in an experiment automated testing relieves expert testers from executing the same monotonous regression test suite again and again. Hence more resources (expert testers) are available for other hard testing activities which were not possible with manual testing. They also stated that timely maintenance of automated test suits is also required and this maintenance is a major drawback of automated testing.

Hoffman in[15] well defined a trade-off between cost and benefit of test automation by considering various ROI (return on investment) factors. From the calculations it is apparently reasonable to conclude that return from the automation are usually seen in the next release that uses it i.e. running and re-running the automated test cases yields considerable savings. Kasurinen et al. in[16] analyzed the industrial applications of the automated testing. They concluded with the findings that nearly 26% of the test cases are automated in industry, adoption of test automation is a demanding effort in software industry and test automation is most commonly used for quality control and quality assurance. Alan et al. in[17] discussed the benefits, challenges and applications of test automation in context of various particular domains like web applications, sensor networks and mobile phone applications etc. Bret in[18] stressed on adopting a development process similar to the development of standard software. They identified different criteria for selecting test cases for automation.

Li et al. in[19] stated that in spiral development, testing cycles are responsible for creating a demand of tools that automate testing. They focused on the design of the tools so as to produce high-quality software in shortened time. Being a step ahead, Damm et al. in[20] describes how test-driven development is possible in industry with test automation. They presented an approach for early detect detection in a cost effective way with the use of C++ to write both software and test tool. Amannejad et al. in[21] automate the integral testing process, ranging from test-case design, to test scripting, to test execution and finally test-result evaluation, in an oil and gas industry and found out that effective decisions on test automation may result in more than 100% ROI in ten rounds of test case rerunning.

A lot more work has been done in this area which emphasizes on the importance and benefits of test automation. However no work, to the best our knowledge, has been done to formulate and calculate the direct effect of test automation on cost, quality and time.

3. Problem Description

The cost benefit analysis of test automation and its impacts on the overall quality and schedule of the software is the central problem that arises from the literature review. This problem is of great significance in the scenarios of Continuous Integration (CI)[22,23] where minor changes are incorporated frequently into the software. Optimization of regression testing cycles, in the scenarios of CI is vital because it has a great impact on revenue of the software industry which is about to produce a new release of an old software. Recent trends have shown that automation of regression testing (fully or partially) is the key to its optimization. And to optimize this particular wing of testing we need to reckon certain parameters in terms of its impact on the time, cost and quality of the software arising out of automated testing. Although a lot of research has been done in the past to calculate the ROI from test automation but none of the author has analyzed the impacts of test automation on these parameters separately. In the next sections we have shown, how to calculate and quantify the consequences of test automation on cost, quality and time to market of the software.

4. Proposed Solution

4.1. Experimental Setup

To calculate the effects of test automation on software's time to market, cost and quality, we set up an experiment over three different of software's. All the software's are produced using iterative enhancement model[24]. In all these software's, after an initial release, subsequent versions of the software's are developed with more added functionality. The details of each of the software used in our experiment are as follows:

4.1.1. Railway's Cloakroom and Retiring Room Management:

This software[†] is prepared for the digital management of Railway's cloakroom and retiring rooms available for the passengers. It is intended to be installed on kiosk for all the passengers having a valid PNR (Passenger Name Record). Passenger can acquire a lock on any available free locker based on his/her choice (available choices are small, medium or large lockers). This software is produced in five versions with the following details: In version 1 only small lockers were available. In version 2, small and large lockers were available. In version 3, small, medium and large lockers were available. In version 4, a bug was found in the previous systems in the code fragment of release lock; that bug was corrected by adding a new error message. In version 5, the railway system demanded a

[†] Available online at https://github.com/kumardivya/test_automation_cloakroom

new software with the functionality of retiring rooms. So while keeping the old system intact, a new functionality to assign and release retiring rooms was also added. UI interface, format and outputs were also changed.

4.1.2. Restaurant Billing System:

This software[‡] was created for restaurants, for placing orders and generating corresponding bills. However as the restaurant grows, it was required to add more type of food items in the menu and more functionalities by the administrator. The version specific information of is software is as follows: Version 1 is designed for restaurants which have "one type of food and only one item under food category". Here you can create order and generate bill against the order. Version 2 is same as version 1 with extra items under one category of food. Version is same as version 2 with various types of food and various items under each food category. Version 4 is same as version 3 with the extra feature of editing the order and editing the quantity if items. Version 5 is version 4 with the features of availability of food item, i.e., if any item is unavailable then ordering of that item is not possible.

4.1.3. Mini Geometric Figure Analyzer

This software[§] is used for simplifying the line-point relations in the geometrical figures. The version specific information of is software is as follows: Version 1 is the initial version and is developed to identify the type of triangle using length of edges of given figure. Version 2 is developed to identify the type of triangle or quadrilateral using length of edges of given figure. In version 3 we can identify the type of triangle using length of edges or coordinates of given figure. As done in previous version, it also finds the type of quadrilateral using length of sides. Version 4 is developed to identify the type of triangle using length of edges or coordinates of given figure. It also recognizes the type of quadrilateral using given coordinates or given length of sides.

4.2. Impacts of Test Automation on Software's Cost

As described in[25,26] we have measured all the cost in terms of effort which has a unit of person/months. So from now on-words cost and effort can be used interchangeably in the forth coming text. The five types of efforts used in our analysis, during any version *i* are: MTE_i*:* Manual Testing Effort, TME_i*:* Automated Test-case Maintenance Effort, ATE_i*:* Automated Testing Effort, TAE_i*:* Test Automation Effort, TTE_i*:* Total Test-team Effort, *STE:* Combined, total Software Testing Effort for all versions. Manual Testing Effort in any version is calculated using Use Case Point (UCP) approach as described by Suresh in[27]. When we are trying to automate a test case we are creating a whole new program, and the cost of developing this whole new software equals Test Automation Effort which can be calculated using Boehm's COCOMO-2[28] model of software cost estimation. This software is relatively small, can be developed by a sub-team of testers or developers and have a set of less rigid requirements hence this can be treated in the category of organic projects of COCOMO-2 with nominal value of cost drivers. To visualize the cost impacts, let us first suppose that there is no test automation in the industry. In this situation the total test-team effort will be equal to manual testing effort and total software testing effort will be a summation of this total test-team effort over all *k* versions of the software.

In the scenario of full test automation, MTE_i is calculated as given in equation 1 but only for the newly developed features. For testing old features in the current version, we will be using test-cases which were automated in the previous version. ATE_i is taken as negligible as the whole testing work is done by machine. TAE_i is calculated through COCOMO-2 model for organic projects as in equation 4 and it is assumed that *30%* of effort is utilized in maintaining the previously written automated test case. Finally net cost impact can be calculated as the difference in $STE_{with\ automation}$ and $STE_{without\ automation}$.

$$MTE_i = AdjustedUseCase\ Points\ in\ Version\ i \quad (1)$$

[‡] Available online at https://github.com/kumardivya/test_automation_restaurant_billing/

[§] Available online at https://github.com/kumardivya/test_automation_geo_calculation/

$$TTE_{i\ (without\ automation)} = MTE_i \tag{2}$$

$$STE_{without\ automatuon} = \sum_{i=1}^{k} TTE_{i\ (without\ automation)} \tag{3}$$

$$TAE_i = 2.4(KLOC)^{1.05} \tag{4}$$

$$TME_{i+1} = 0.3(TAE_i) \tag{5}$$

$$TTE_{i\ (with\ automation)} = MTE_i + (TME_i + ATE_i) + TAE_i \tag{6}$$

$$STE_{with\ automation} = \sum_{i=1}^{k} TTE_{i\ (with\ automation)} \tag{7}$$

$$Cost\ or\ Effort\ Impact = STE_{without\ automation} - STE_{with\ automation} \tag{8}$$

4.3. Impacts of Test Automation on Software's Quality

Software testing has a direct bond with software quality. Software's quality is the level to which it coheres with the laid specifications[29] and in the testing phase we are measuring how much the software adheres to its requirements. To perform a quantitative analysis of software's quality ISO 9126-1[30] have defined six main quality characteristics that are: Functionality, Reliability, Usability, Efficiency, Maintainability and Portability. Out of these features Functionality (degree to which program fulfills its requirements), Reliability (extent of failure free operations of software during a specified time slot) and Maintainability (time and effort required to analyze a failure in an operational software, to change it and then test the changed system) prominently affect the quality of the software. The others characteristics which are Usability, Efficiency and Portability are ignored in our analysis as these characteristics and their measurements are isolated from the testing phase. The focused features i.e. Functionality, Reliability and Maintainability are calculated according to the equation 9, 10 and 11 respectively.

$$Functionality\ F = \frac{number\ of\ SRS\ features\ fulfilled}{total\ number\ of\ features\ required} \tag{9}$$

$$Reliability\ R = \frac{mean\ time\ to\ failure}{mean\ time\ to\ failure + mean\ time\ to\ repair} \tag{10}$$

$$Maintainability\ M = \frac{time\ spend\ on\ testing}{total\ development\ time} \tag{11}$$

4.4. Impacts of Test Automation on Software's Time

There is a direct correlation between time and effort[31–34]. The impacts of test automation on the total time to market of the software can be calculated simply by adjusting the gained or lost effort from equation 8. We have used the COCOMO-2's second equation for calculation the development time based on the required effort. Although this equation is used to calculate the total development time needed for the production of whole software, this equation is used for the effort to time conversion also. In our calculations we have used the second formula of COCOMO-2 for organic type of projects and nominal value of cost drivers, as given in equation 12. The reader may use other value of COCOMO-2 coefficients c, d, if their projects are semi-detached or embedded type.

$$Time\ Impact = 2.4(Effort\ Impact)^{0.38} \tag{12}$$

Table 1: Cost Quality and Time Impacts of Test Automation

Software	Version	Cost and Time Impact						Quality Impact					
		Without Automation	With Automation					Without Automation			With Automation		
		AUCP=MTE=TTE=STE	MTE	KLOC	TAE	TME	STE	F	R	M	F	R	M
Software 1: Railway's Cloakroom and Retiring Room Management	1	39	39	1280	3.11	0	42.11	4	0.7	0.61	3	0.9	0.55
	2	45	8	1080	2.60	0.93	11.53	3	0.9	0.69	4	0.7	0.65
	3	52	8	1077	2.59	0.73	11.37	5	0.5	0.77	3	0.9	0.64
	4	60	11	1233	2.99	0.77	14.76	6	0.4	0.74	5	0.5	0.52
	5	85	17	1486	3.63	0.89	21.53	6	0.4	0.72	3	0.9	0.51
	Total	281					101.32						
						Cost Impact	179.67						
						Time Impact	3.045						
Software 2: Restaurant Billing System	1	35	35	1430	3.49	0	38.49	2	0.6	0.67	3	0.9	0.57
	2	55	21	1365	3.32	1.04	25.37	3	0.6	0.68	3	0.7	0.61
	3	50	15	1500	3.67	0.99	19.67	2	0.7	0.60	1	0.9	0.63
	4	60	15	1400	3.41	1.10	19.51	4	0.7	0.64	3	0.8	0.61
	5	70	15	1600	3.93	1.02	19.95	1	0.8	0.56	1	0.8	0.51
	Total	270					123.01						
						Cost Impact	146.98						
						Time Impact	3.022						
Software 3: Mini Geometric Figure Analyzer	1	50	50	1286	3.12	0	53.12	5	0.6	0.66	3	0.9	0.55
	2	51	10	1968	4.88	0.93	15.82	4	0.9	0.68	4	0.9	0.64
	3	66	14	1456	3.56	1.46	19.02	3	0.9	0.76	3	0.9	0.63
	4	68	10	1589	3.90	1.06	14.97	5	0.8	0.71	3	0.9	0.52
	Total	235					102.94						
						Cost Impact	132.05						
						Time Impact	3.009						

5. Results

For all the software projects deployed in our study, there are all around favorable impacts of software test automation. Table 1 shows the impacts of test automation on three software's i.e. Railway's Cloakroom and Retiring Room Management, Restaurant Billing System and Mini Geometric Figure Analyzer. The effect of test automation is measured along the all versions of these software's. All It is a common observation in all the projects that there is a positive cost and time impacts of test automation and quality is also improved in most of the cases as program is found incorrect fewer numbers of times with automated test cases than with manual testing. The availability increases in all the cases and relative time in testing is also fairly decreased because of test automation.

6. Conclusion and Future Work

Software testing has a prime importance in software's verification and validation. It is important because of two main reasons, first, it assures software quality, and second, nearly 60% of the total software's cost is spend over different types of testing. Regression testing is the object of interest in this paper and we have detailed the effects of automation of regression tests over the software's cost, quality and time to market. In our experiments in we have used three different software's which are, developed in versions. We have formulated mathematical models, based on the various significant testing effort factors, to quantify the impact of test automation. Although, automation of test cases have a high implementation and maintenance costs, from our experiments we have found that, automation of test cases can give remarkable returns in the long runs where we run and rerun the automated-tests, multiple times. We have also found that test automation has positive effects on software quality. Hence we can claim that test automation increases the overall effectiveness of the testing process when we have repetitive testing tasks which are similar. This work can be extended in future by adding more variable automation cost factors in the analysis to make it more precise and accurate. The concept of automatic test data generation can also be combined with the present research.

References

1. Kit, E. & Finzi, S. *Software Testing in the Real World: Improving the Process*. (ACM Press/Addison-Wesley Publishing Co.: New York, NY, USA, 1995).
2. Myers, G. J., Sandler, C. & Badgett, T. *The art of software testing*. (John Wiley & Sons: 2011).
3. Oster, N. & Saglietti, F. Automatic test data generation by multi-objective optimisation. *SAFECOMP* **4166**, 426–438 (2006).
4. Ramler, R. & Wolfmaier, K. Economic perspectives in test automation: balancing automated and manual testing with opportunity cost. *Proceedings of the 2006 international workshop on Automation of software test* 85–91 (2006).
5. Jalote, P. *An integrated approach to software engineering*. (Springer Science & Business Media: 2012).
6. Hetzel, B. *The Complete Guide to Software Testing*. (QED Information Sciences, Inc.: Wellesley, MA, USA, 1988).
7. Dustin, E., Rashka, J. & Paul, J. *Automated software testing: introduction, management, and performance*. (Addison-Wesley Professional: 1999).
8. Hoffman, D. Test automation architectures: planning for test automation. *Quality Week* 37–45 (1999).
9. Page, A., Johnston, K. & Rollison, B. *How we test software at Microsoft*. (Microsoft Press: 2008).
10. Pettichord, B. Seven steps to test automation success. *STAR West, San Jose, NV, USA* (1999).
11. Dustin, E. *Effective Software Testing: 50 Ways to Improve Your Software Testing*. (Addison-Wesley Longman Publishing Co., Inc.: 2002).
12. Bertolino, A. Software testing research: Achievements, challenges, dreams. *2007 Future of Software Engineering* 85–103 (2007).
13. Stobie, K. Too much automation or not enough? When to automate testing. *Pacific Northwest Software Quality Conference* (2009).
14. Berner, S., Weber, R. & Keller, R. K. Observations and lessons learned from automated testing. *Proceedings of the 27th international conference on Software engineering* 571–579 (2005).
15. Hoffman, D. Cost benefits analysis of test automation. *STAR West* **99**, (1999).
16. Kasurinen, J., Taipale, O. & Smolander, K. Software test automation in practice: empirical observations. *Advances in Software Engineering* **2010**, (2010).
17. Hartman, A., Katara, M. & Paradkar, A. Domain Specific Approaches to Software Test Automation. *The 6th Joint Meeting on European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering: Companion Papers* 621–622 (2007).doi:10.1145/1295014.1295062
18. Pettichord, B. Success with test automation. *Ninth International Quality Week, Software Research, San Francisco* (1996).
19. Li, K. & Wu, M. *Effective software test automation: developing an automated software testing tool*. (John Wiley & Sons: 2006).
20. Damm, L.-O., Lundberg, L. & Olsson, D. Introducing Test Automation and Test-Driven Development: An Experience Report. *Electronic Notes in Theoretical Computer Science* **116**, 3–15 (2005).

21. Amannejad, Y., Garousi, V., Irving, R. & Sahaf, Z. A Search-Based Approach for Cost-Effective Software Test Automation Decision Support and an Industrial Case Study. *Software Testing, Verification and Validation Workshops (ICSTW), 2014 IEEE Seventh International Conference on* 302–311 (2014).doi:10.1109/ICSTW.2014.34
22. Fitzgerald, B. & Stol, K.-J. Continuous software engineering and beyond: trends and challenges. *Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering* 1–9 (2014).
23. Turk, D., France, R. & Rumpe, B. Assumptions underlying agile software development processes. *arXiv preprint arXiv:1409.6610* (2014).
24. Larman, C. & Basili, V. R. Iterative and incremental development: A brief history. *Computer* 47–56 (2003).
25. Kemerer, C. F. An empirical validation of software cost estimation models. *Communications of the ACM* **30**, 416–429 (1987).
26. Pressman, R. S. *Software engineering: a practitioner's approach*. (Palgrave Macmillan: 2005).
27. Nageswaran, S. Test effort estimation using use case points. *Quality Week* 1–6 (2001).
28. Boehm, B. W., Madachy, R., Steece, B. & others *Software cost estimation with Cocomo II with Cdrom*. (Prentice Hall PTR: 2000).
29. Kitchenham, B. & Pfleeger, S. L. Software quality: The elusive target. *IEEE software* 12–21 (1996).
30. Iso, I. IEC 9126-1: Software Engineering-Product Quality-Part 1: Quality Model. *Geneva, Switzerland: International Organization for Standardization* (2001).
31. Agrawal, M. & Chari, K. Software effort, quality, and cycle time: A study of CMM level 5 projects. *Software Engineering, IEEE Transactions on* **33**, 145–156 (2007).
32. Boehm, B. W. & others *Software engineering economics*. **197**, (Prentice-hall Englewood Cliffs (NJ): 1981).
33. Capra, E., Francalanci, C. & Merlo, F. An empirical study on the relationship between software design quality, development effort and governance in open source projects. *Software Engineering, IEEE Transactions on* **34**, 765–782 (2008).
34. Lind, R. K. & Vairavan, K. An experimental investigation of software metrics and their relationship to software development effort. *IEEE Transactions on Software Engineering* 649–653 (1989).