A System of Humanizing Test Automation Outlay Efficiency

Saket Vihari, Arun Prakash Agrawal

Abstract- Software can be tested either manually or automatically. The two approaches are complementary: automated testing can perform a large number of test in little time, whereas manual testing uses the knowledge of the testing engineer to target testing to the part of the system that are assumed to be more error-prone. Auto test is a testing tool that provide a "best of both worlds" strategy :it integrates developers test cases into an automated process of systematic contract –driven testing. Test automation has become more and more popular as the market demand for more complex software, involving higher risks and using the same or fewer resources in development, has increased. A number of research paper discuss the problem faced in the test automation process ,such as the complexity of automation ,poor choice of tools,and the effort spent to automate. This paper proposes a test automation viability analysis method of a test case based on mathematical procedures which intend to increase the chance of finding outlay efficiency test automation process.

Index Terms— Test automation ,outlay efficiency, Viability, Method.

I. INTRODUCTION

An automated testing strategy tries to remove the tediousness of the process by relying on a software tool that generates test cases from the program's specification (black box) or its actual text. Recent advantage in technology have more complex and riskier application, which in turn generates a need to improve software quality. To this end, together with the need to find bugs faster at minimum cost, many organizations have invested part of their project budgets in software test automation. Test automation has thus become more and more popular in recent years and has been a constantly increasing activity in the existing software industry.

The idea of having a computer run test instead of running them manually has led many organization to attempt test automation without a clear understanding of all that is involved . On the one hand, test automation can bring several advantages: it is a way of getting more done with less time and fewer resources; the tests can be rerun many times without overheads too much effort, which makes it easier to find bugs earlier and fix them more cheaply; the results appear more reliable. On the other hand, to make a successful test automation process, a large investment of time and expertise is required, without which the process could be destroys.

As a result, many attempts at test automation have failed to achieve real or lasting benefits [1].

Saket Vihari, M.Tech(CSE), Amity School of Engineering and Technologu, Amity University, NOIDA, Uttar Pradesn, India.
Arun Prakash Agrawal, Asst. Prof., Amity School of Engineering and Technologu, Amity University, NOIDA, Uttar Pradesn, India.

The first idea of many company that want to have their software tests automated is tottery to automate all the tests in order to have them being executed quickly in the minimum amount of time. However, it is always good to keep in mind that the investment needed to automate the testing process is very high, and sometimes the outlay efficiency of automating everything can be too low.

The goal of this paper is to offer a feasibility analysis method, to help testers decide which tests can be automated cost efficiency. An example of how to use the method is also presented to demonstrate how the proposed feasibility analysis method work.

II. TEST AUTOMATION

Manual testing is performed by a human sitting in front of a computer carefully executing the test steps. Automation Testing means using an automation tool to execute your test case suite. The automation software can also enter test data into the System Under Test , compare expected and actual results and generate detailed test reports.Test Automation demands considerable investments of money and resources. Successive development cycles will require execution of same test suite repeatedly. Using a test automation tool it's possible to record this test suite and re-play it as required. Once the test suite is automated, no human intervention is required . This improved ROI of Test Automation .Goal of Automation is to reduce number of test cases to be run manually and not eliminate manual testing all together.

Automated testing is important due to following reasons:

- Manual Testing of all work flows, all fields , all negative scenarios is time and cost consuming.
- It is difficult to test for multi lingual sites manually
- Automation does not require Human intervention. You can run automated test unattended (overnight).
- Automation increases speed of test execution.
- Automation helps increase Test Coverage.
- Manual Testing can become boring and hence error prone.

III. PRE-ANALYSIS

In the last few years, automation has become one of the main investments in organizations in order to improve their software quality. At Borland, for example, after a significant investment in automation, only 20% of software bugs were found by automated est cases. Borland claims that manual tests were "more variable and more directed at new features and specific areas of change where bugs were more likely to be found" [2].

Even with the current low level of practical success of test automation, it is strange to see how large the attraction is to it [3].

According to the International Institute of Software Testing, only15% of all test automation initiatives succeed [4]. Before starting the automation process, two analyses need to be carried out. First, it is necessary to check which tests are technically possible to be automated. Second, it is necessary to verify if the tests identified in the first analysis are viable to be automated. A test can be automated if, with all available technology, it could give the same result as if it were executed manually. Considering a test automatable does not mean we should automate it. There are various reasons why a test case should not be automated without a careful previous analysis in order to have a successful automation project. "Making good decisions about what to automate can be critical to successful test automation" [5].This is the goal of this paper. A mathematically generated decision tree is being proposed to carry out a viability analysis in order to know if a single test is or is not a candidate for automation. The analysis is based on nine topics that will be detailed one by one in the next section, to expose their importance. After this, the feasibility analysis method will be explained and an example of how to put it into practice will be described.

Implementation Frequency

One of the important points to reach through the Feasibilty analysis is concerned with how many times a specific test case is going to be executed. If a test case is going to be executed only once, the automation of this test case may be completely useless. A good practice is to compare the effort spent to automate and the effort spent to execute the test manually to see how many executions it takes to achieve gains by carrying out the automation process. If the number of executions is less than the number that found, automation might not be a good option.

Making of Reusable Code

When a test case is automated, it is supposed to give some contribution to the framework used to automate it. Even a very complex test case can be a good candidate for automation, if it contributes with important new features to the framework/library, that is, when the code used in that specific test case will be reused in other test cases. This point becomes even more important when the execution frequency of the test case under analysis is low since, at first glance, the effort might not be rewarded. However, if the code is reusable, even if the test case itself is not executed frequently, the tester will benefit greatly when the code produced is used in the future.

Test Importance

The number of bugs the test case is supposed to find is another point to be considered. The aim of any test case is to find bugs, but some test cases have a lot more relevance than others, since they can test critical functionalities or even functionalities that will be used more often than others. For example, in the context of a bank web site, the tests involving a login page would be much less relevant than the tests related to account transactions. However, they might achieve the same relevance because the frequency of a login operation is higher than that of account transactions.

Automation Effort

The effort spent to automate a single test case must be considered carefully before starting the actual process. The total effort spent on the whole automation process can annul almost all of the automation process' advantages [6]. There are only a few reasons to keep thinking of automation for a test case that takes too much effort to be automated, such as: the test lifetime and frequency of execution, and the reusability of the piece of code that this test case produces to develop other test cases. These factors must be well analyzed.

Resources

It is important to know the cost to deploy the test case. A test case may need some brand new technology or high-performance hardware to be automatable, and it may cause high or extra costs to the company. One should consider which is more profitable or provides the least losses: buying more equipment or taking on more people for your team. Another important point to be considered is about how many people in your team are necessary to execute a single test case. Furthermore, when there is a cycle of test cases to be executed, one might need to know how many members must be allocated to perform the whole test cycle. If these tests are automated, only one person is required to perform this action.

Manual Complexity

In many applications, the training costs to make testers available to run a single test case is too high when compared to having this test automated. Test cases that require a lot of special knowledge appear as very good candidates for automation since not anyone could test it manually while, with an automated test, any tester is

able to carry it out [1]. Considering another perspective, some test cases imply directly in revealing confidential information to everybody that will possibly execute them. The information that can be learned with the execution of a test case may be important somehow to the knowledge of the test team. If the complexity of manual execution becomes an issue in any other way, automation will be a good answer.

Automation Tool

The automation tool must be very carefully chosen before the test automation process begins. The tester must know it deeply to be able to differentiate an SUT (software under test) bug from an automation tool bug. Having reusable functions or class libraries is essential to obtaining a good automation process. Complex functions that might not have the necessary trustworthiness because of an automation tool's dependencies must be well considered before being created. This seems like simple advice, but it's a very difficult issue to solve: wrong results reported by the tool. The code must be as reusable and portable as possible for the whole suite of platforms to test.

Interface

Changes in the environment where the tests are being run might cause a lot of rework on the test automation framework, or even on the existing scripts, which were made for a specific environment. One of the biggest challenges to using automated test suites is keeping them functional as the product interface changes.

What if a test case is implemented and performed a number of times, but its requirements change and the test case needs to be performed in another environment? In fact, before deciding to automate a test case it is necessary to know how static the environment is. If the environment changes, it is important to be prepared for it.

Building a test case that can be ported to as many environments as possible, or which predicts low costs to be re implemented to another environment, is very good practice.

Implementation Effort

The effort spent while a test is being run is a variable to be considered in our automation viability analysis method. It is good when a test case execution effort can be compared in its manual and automated execution [5]. If an automated test runs faster than running it manually, things would be relatively simple, but unfortunately this does not occur every time. Sometimes, the automatic execution of the test is slower than the manual execution of the same test case. However, this is no reason to give up automating some test cases. Considering a manual test that runs faster than the automated one, one might decide not to use automation, but even in this case an automated test can be performed better than its manual equivalent. This test case can probably be performed with other tests engaged, so a lot of tests can be run in a test cycle at once. If the automated method is chosen, the test can run all night.

IV. FEASIBILITY ANALYSIS METHOD

On The basis of topic discussed in the previous section , some question were proposed whose answer will be analysed and judged properly to get an indicator of the outlay efficiency of each test case ,as illustrated in following table:

Table 1: Question For Each point

[1] Identifier	[2] Topics	[3] Related Questions
[4] 1	[5] Frequency	[6] Ow many times is this test supposed to be executed?
[7] 2	[8] Reusable [9] Code	[10] Can this test or part of it be reused in other test?
[11] 3	[12] Importance	[13] How would you describe the importance of this test case
[14] 4	[15] Automation [16] Effort	[17] Does this test take a lot of effort to be displayed ?
[18] 5	[19] Resources [20]	[21] How many member of your team should be allocated or how expensive is the equipment needed during this test"manual Execution?
[22] 6	[23] Manual Complexity	[24] Is this test Difficult to be implemented manually? Does it have any embedded confidential information?
[25] 7	[26] Automation Tool	[27] How would you describe the reliability of the automation tool to be used ?
[28] 8	[29] Interface	[30] How much interfacable is this test?
[31] 9	[32] Implementation Effort	[33] Does this requires a lot of effort to be implemented Manually

The questions presented in Table 1 were answered for 500 previously automated test cases to serve as input for the Decision Tree Learning Algorithm [8]. This set of test cases involves two different levels of testing (system and integration) and three different types of test (GUI, performance and stress). Figure 1. The process of generating and validating the tree is automatic. After receiving 500 entries to generate the tree, it was validated with 200 different entries. The validation was done using test cases in integration and system levels. The results obtained were Compared with the manual results, which gave us an average assertion of 85.5%, as shown in Table 3. Since a good assertion percentage result was obtained, the tree was eligible for use in the automation viability analysis method. Table 2 presents the classification of the 500 test cases, showing how many test cases were used from each level and from each type of test. The Decision Tree Learning Algorithm was implemented and the inputs generated beforehand were supplied. The system learns with the entries that are offered to it and suggests a model to be used.

The model suggested with the 500 inputs generated was the decision tree illustrated in Figure 1. The process of generating and validating the tree is automatic. After receiving 500 entries to generate the tree, it was validated with 200 different entries. The validation was done using test cases in integration and system levels. The results obtained were compared with the manual results, which gave us an average assertion of 85.5%, as shown in Table 3. Since a good assertion percentage result was obtained, the tree was eligible for use in the automation viability analysis method

Table2: Number Of test Cases by test level and test types

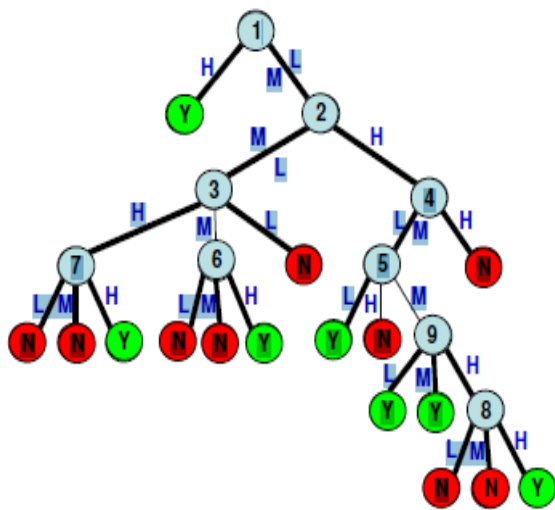| Test level / Test type	System test	Integration test
Performance	60	50
GUI	220	70
stress	60	40

Fig 1: Decision tree for feasibility analysis

The tree represents the ways that can be followed by answering the questions presented in Table 1. The numbers in the grey circles are the Identifiers for the questions that they are related to. The 'Y' and 'N' represent the end of the tree and mean 'Yes' or 'No', respectively. They are the indicators of outlay efficiency.

Table 3: validation results

Test Level	Number of Test cases Used	Assertion Number	Assertion Percentage
System	150	127	85%
Integration	50	43	86%
Total	200	170	85.5%

All the proposed questions have a discrete number of answers: 'High', 'Medium' or 'Low', which are represented in the tree by the letters 'H', 'M' and 'L'. Note that, depending on the answer of each specific question, the tree takes you different ways. Going through the tree from the top to the end via the answered questions, an indicator of 'Yes' or 'No' will be obtained, showing if a test case is viable or not for automation. It is important to notice that not all the questions need to be answered for a test case. For example: if the answer given to the question number 2 is 'H', question number 3 will never be answered. Also, it is worth clarifying that there is no correct answer for these questions. However, the more you know about the tests being analyzed, the better the chances will be for success.

V. USING THE METHOD

To better understand the Decision Tree and how it works, we provide an example. First, a scenario was created as shown in Table 4. This presents the test case generated for the scenario described before.

Table 4: example- Online Trading

[34] Scenario: In online trading on the internate , the user should be able to perform a large amount of transaction such as money transfer, product selling and purchasing, payments among others .

Table 5: Test Case

Test case Description : Verify if an amount of money is debited from an account "A" and Credited to an account "B" when a user tries to transfer money from A to B		
Steps	Action	Expected Result
1	Client successfully log into the system to account A	A list with all available transaction is shown
2	Client chooses to perform a payment	The field related to agency and account number and the values to be transferred are prompted to the user
3	Client fills in field to payment the money to account B	If the information is valid , the password will be requested. If the information is not valid ,user will be prompted with an error message and will be asked to try again.
4	Client types his password and confirm the operation	The previously typed amount is debited from Account A and credited to account B

Starting with the first question on the tree, "How many times is this test supposed to be executed?", if this operation has few executions, the answer is 'Low'. This answer takes you to the right side of the tree, leading to the second question, "Can this test or parts of it be reused in other tests?" Let's suppose that the code used to automate this test has little chance of being reused. Thus, the answer to Question 2 is 'Low'. Now the Decision Tree takes us to the left side of the tree, to the next question, "How would you describe the importance of this test case?" Making a transaction on a bank website is an important task to be tested, so the answer is 'High'. The left side is taken, which leads to the last question, "How would you describe the reliability of the automation tool to be used?" As the test has very high relevance, the tool to be used must be quite reliable to ensure that this test is in fact being well executed. Therefore, the answer to this question is 'High'. The summary of the results reached with this example is presented in table 6.

Table 6: Answer for the example

[35] Identifier	[36] Question	[37] Answer
[38] 1	[39] How many times is this test supposed to be executed ?	[40] Low
[41] 2	[42] Can this test or parts of it be reused in other test ?	[43] Low
[44] 3	[45] How would you describe the importance of this test case ?	[46] High

[47] 7	[48] How would you describe the reliability of the automation tool to be used ?	[49] High

By answering the questions 1, 2, 3 and 7, following the decision tree, the user would have a positive response, which would mark this test as a good candidate for automation. Note that it is not necessary to answer all the questions. Depending on the answers that are given, the tree can conduct the user to answer only some of the questions.

VI. CONCLUSION

There are other studies that show the problems experienced in a software test automation process, as presented in [1], [6] and [3].The current study differs from those because it not only points out the problems, but also proposes a new way of dealing with these problems. It offers a method to analyze the tests, choosing which test cases are viable for automation before starting the automation process. Since the success rate of the automation process is low[7], the work presented here can increase the chances of having a outlay efficiency process. The study of the feasibility analysis method was based on a mathematical procedure and the experience of the team, which increases the trustworthiness of the work done.

According to the algorithm, the decision tree proposed also has the ability to change as more answers are provided. This process is an evaluative way of making decisions. However, the tree used in the method proposed was constructed using a high number of different entries, and it was noticed that when it reached arround 300 entries the tree looked very stable, suffering only small changes with the last 200 entries.

For future work, more experiments can be done to increase the trustworthiness of the tree. It can be trained with other types of tests and also validated with other entries to analyze other sets of results and compare them with the current ones.

To improve the tool that is being built, other studies are being done in order to define a scale of how easily the test case can be automated. Based on this scale, and using methods such as neural networks, committee machines and MLP, an order in which the test cases should be automated will be suggested, going from the easiest to the most difficult ones. Developing the test cases following a pre-defined order will help automation agility since the code can be reused, contributing to the improvement of the framework library.

REFERENCES

1. M. Fewster, "Jumping Into Automation Adventure with Your Eyes Open", Journal of Software Testing Professionals, March, 2002.
2. J. Bach, "Test Automation Snake Oil", presented at 14th International Conference on Testing Computer Software, Washington-USA, 1999.
3. J. Kent, "Advanced approaches to Software Test Automation Part 1", Journal of Software Testing Professionals, June, 2001.
4. J. G. Soderborg, "Five Factors for Finding Test Automation Success", WEB Seminars, available in http://www.segue.com/aboutsegue/webinars/public/sdtimes_en/index.html, April, 2005.
5. B. Pettichord, "Success with Test Automation", presented at Quality Week, San Francisco, May, 1996.
6. B. Marick, "When Should a Test Be Automated", presented at Quality Week' 98, San Francisco, 1998.
7. L. Hayes, "Establishing a Test Automation Function", Journal of Software Testing Professionals, March, 2000.
8. S. Russel and P. Norvig, Artificial Intelligence, Prentice Hall; 2nd edition (December 20, 2002), USA, December 20, 2002.

AUTHORS PROFILE

Mr. Saket Vihari is Presently working in the National Physical laboratory(Council of scientific and Industrial research) , Newdelhi as a Technical staff . He has got his MSc in Electronics From Bihar University in 2008. He has more than four years of Experience In different Electrical , electronics and Material Testing Laboratory. He has persuing his M.Tech (CSE) from Amity University Noida. Currently His area of research is dedicated to Automation testing of different system.

Arun Prakash Agrawal is Asst. Prof. in Amity University,NOIDA,UP.He has ten years of experience of teaching at graduate and post-graduate levels. He obtained his M.Tech. in Computer Science and Engineering from Guru Gobind Singh Indraprastha University, Delhi. He is also the gold medalist of his batch. He has taught at various engineering colleges of repute. Presently he is pursuing his Ph.D. in the area of software engineering from Guru Gobind Singh Indraprastha University, Delhi. He has several research papers of national and international repute to his credit. He He is a member of IEEE computer society, IAENG and ACM. His area of interest includes software engineering, software testing, computer networks and mobile computing.