THE DEVELOPMENT OF RELIABLE ESTIMATION METHODS FOR THE USAGE OF TEST AUTOMATION IN AN AGILE ENVIRONMENT

Short Paper

Jos van Rooyen
Identify Test Services BV
Rosmalen, the Netherlands
jos.van.rooyen@identify.nl

***Abstract*- One way to ensure the quality of software is by testing it. Running tests can be done both manually and automatically. Test automation, with its ups and downs, has been the centre of attention for many years. It is usually underestimated what implementing test automation entails and the impact it has on an organization, especially the estimation of the required effort. Adding test automation within the entire range of testing measures requires extra human capacity, both for the initial set up and the maintenance of the automatized tests apart from test implementation. The question is how much human capacity is needed in order to test automatize the functionality which has to be tested automatically? This article describes several methods to estimate the required effort for test automation and the approach to collect the required data. By applying the described estimation methods via a case study key figures can be defined to estimate the required effort for test automation projects.**

Keywords-Estimation; Test Automation; Agile; Testing; Return on Investment; Future proof.

I. INTRODUCTION

During Valid2016 the first ideas were presented [1] how to estimate test automation in an Agile environment. Based on the discussion during the conference and further elaboration of the topic, this paper describes several methods to estimate the required effort for test automation. Inside testing [2], test automation is one way to ensure the quality of software.

What is meant by testing software is the following [3]:

"The process consisting of all lifecycle activities, both static and dynamic, concerned with planning, preparation and evaluation of software products and related work products to determine that they satisfy specified requirements, to demonstrate that they fit for purpose and to detect defects."

Running tests can be done both manually and automatically. Test automation, with its ups and downs, has been the centre of attention for many years. It is usually underestimated what implementing test automation entails and the impact it has on an organization [4][5]. Because of the rising popularity of Agile [6] and the implementation of continuous deployment and development [7], it seems that test automation is taking on a fixed position. The most important reason for this is that the amount of work is no longer manageable to be done manually [8].

Adding test automation within the entire range of testing measures requires extra human capacity, both for the initial set up and the maintenance of the automatized tests apart from test implementation. The question is how much human capacity is needed in order to test automatize the functionality which has to be tested automatically?

Test budgeting has been a problem since the beginning [9]. Several methods have been developed [10][11][12], but they do not always produce the correct results. Practice shows that significantly more time is needed than was budgeted at the start of the project.

The question is how to get a grip on this in order to make reliable predictions concerning the necessary capacity. Based on previous methods [10][11][12], which are the utilized ways of budgeting within the Agile methodology [13], three ways of thinking have been developed to budget automatized test capacity. These three ways of thinking are described in this article. However, they still have to be tested in practice.

The fundamental principle in this article is a structural, future proof design of test automation within the Agile developed methodology. That means the following: designing test automation in such a way that developed automatized tests cannot only be executed, reused, and easily transferred to others today, but also in the future, and done in such a way that maintenance effort is minimal.

The paper has the following structure. Section II describes the causes of poor test budgets on behalf of test automation. Section III describes the general elements that affect the required test capacity. Section IV will give insight into budgeting future proof elements on behalf of test automation. Section V discusses the three ways of budgeting. A detailed example has been included in Section VI. Section VII describes the collection of the data and the approach to classify into the described estimation methods. Return on investment is dealt with in Section VIII. Lastly, Section IX describes the conclusions and future work.

II. Causes of poor test automation budgeting

As indicated in the introduction, budgeting within ICT is a common problem [9]. Which causes are at the core of this? A couple of reasons can be found.

Using new development and or programming techniques of which there is not enough knowledge. Not questioning the desired functionality enough which causes new problems to arise during the implementation and test phase. Unfamiliarity with the quality of the software in the beginning is another reason. Furthermore, the quality of the persons involved, such as the tester or developer, plays a role. Is someone sufficiently skilled to make a solid budget? [12][11][10].

What we see in practice is that experience numbers are hardly recorded, if recorded at all. This is especially the case for budgeting test automation. A short research in the ISBSG database [14] shows that only three projects have been recorded in which Agile development technique is combined with automatized testing. Of only 1 project out of these three, the delivered test effort has been registered (see Table I).

TABLE I: ANALYSIS ISBSG-DATABASE FOR THE ATTENTION OF TEST projects

#projects	Way of testing		Agile development methodology		Test effort known	
	Manual	Automatized	Y	N	Y	N
95	29	66	3	63	1	2

It becomes clear from this analysis that there is no useful data available to draw conclusions.

In order to try to answer the question: “How is budgeting done in an Agile development environment concerning test automation,” a survey has been conducted in which 100 people participated. The results are recorded in Table II.

TABLE II: RESULTS SURVEY WAY OF BUDGETING TEST AUTOMATION

Way of budgeting	Number
Not budgeted	2
Percentage available time	1
Pokering of the effort	3
Experience based	5
No response	89
Total	100

As Table II shows, no reliable results can be extracted from the survey. Despite a reminder, response was very low. Both the results of the survey and the analysis of the ISBSG database, which show comparable results, were a trigger to keep on thinking of ways how to budget test automation reliably and predictably.

A first draft was made during the Valid2016 conference where the first ideas were drafted during the presentation: “Estimation of test automation in an Agile Environment” in which the following question was discussed: “How to estimate the required effort in an Agile environment regarding test automation” [1].

During this presentation three ways of thinking were sketched how test automation can be budgeted in an Agile development environment in order to set up test automation in a structured and future proof manner. This article elaborates on the presentation whereby received input has been included in further working out the ways of thinking. Besides the manner of budgeting, each approach has a number of general elements which influence the eventual budget for test automation. These general elements will be elaborated on first.

III. General elements influencing budgeting of test automation

Apart from the required budget to automatize the test scripts, there are several preconditional elements which influence the required budget for test automation. No matter at which level in the organization (project, division or company level) [15] you wish to implement test automation, you will have to deal with these elements. Dependent on the level at which you would like to implement test automation, the impact on the organization will be bigger. If you focus test automation on company level instead of a project or individual sprint, the involved elements will have a wider impact. The elements can be separated in the so-called initial costs and continuity costs. The initial costs are those which you have when setting up and developing test automation for the first time in an organization.

Continuity costs are costs which have to be made after the introduction of test automation in order to maintain and expand (if necessary) test automation. In Table III, the relevant elements are mentioned with an indication how these elements can be measured and a short explanation.

TABLE III: INITIAL AND CONTINUITY ELEMENTS TEST AUTOMATION

Component	Element	Unit of measure	Explanation
Initial costs	People	#people Costs per day	Number of people that are going to work on test automation
	Test tools	#tools Price per license	Type and number of test tools to purchase aligned with various development platforms
	Installation costs	#days Costs per day	Installation of test tools in the ICT- landscape
	Test data	#days Costs per day	Choosing a working method: formulate test data requirements, making synthetic test data, scrambling production data to use as test data. Taking privacy into account [16]
	Education	# days	Number of required educations/courses
	Frequency of usage	#runs	How often is test automation used?
	Virtualization		Is virtualization used?

	Number of integrations with surrounding systems	#integrations costs per integration	Which integrations are relevant and how are they mutually dependent?
	Support which company objectives	n/a (not applicable)	Which strategic objectives have to be supported?
Continuity	License costs test tools	Price per license	Annual costs on behalf of the test tools
	Maintenance test scripts	Modification frequency	Percentage time reserved for maintenance of the test scripts
	Additional training	#days #days upgrade	Required training for new employees and new versions of test tools
	Test tool upgrade	#licenses times updates	Costs linked to purchasing and installing test tool upgrades
	Integrations	#integrations costs per integration	Expansion and maintenance of integrations surrounding systems

IV. BUDGETING FUTURE PROOF ELEMENTS

This information is partly delivered by the overarching test procedure on company level. You can think of frameworks for reusability, tooling, test data generation for repeatability and a wiki for setting up the transferability aspect.

The question is which percentage of the required test capacity for test automation has to be reserved for developing future proof automated test scripts?

The following rules of thumb can be applied as shown in Table IV.

TABLE IV: RULES OF THUMB ON DETERMINING FUTURE PROOF FACTOR

Aspect	Priority	Factor
Reusability	H (= High)	1.2
	M (= Medium)	1
	L (= Low)	0.8
Repeatability	H (= High)	1.2
	M (= Medium)	1
	L (= Low)	0.8
Transferability	H (= High)	1.2
	M (= Medium)	1
	L (= Low)	0.8

An example:

100 hours have been calculated for test automation. In order to set up future proof test automation, the following values have been agreed upon (see below). Determining these values is done in accordance with the principal and is linked to a company's objectives.

Aspect	Factor
Reusability	H
Repeatability	M
Transferability	M

The number of hours required to set up this part future proof will be:

(100 x 1.2) x1 x 1 = 120 hours.

Another aspect to take into consideration is the scope of test automation. For which test type [3] are the test automation scripts developed? A sprint, an integration test (IT) or a chain test (CT)? The bigger the scope, the more synchronization with all parties becomes necessary. Think of which test data to use, which test scripts, availability of test environments for example and analysis of the results [17] [18].

You can introduce another factor namely the test type with the following parameters:

Test type	Factor
Sprint	1
IT	1.2
CT	1.5

Say you would like to do for example a chain test automation. The necessary effort, based on the table above would be:

(120 x 1.5) = 180 hours.

As stated, these are rules of thumb, which will have to be tested and adjusted by collecting data from yet to be executed case studies.

V. BUDGETING TEST AUTOMATION

In previous sections, it has been discussed both which general elements influence the test automation budget and that making test automation future proof also impacts the budget.
So how do you really budget test automation?

The following methods of budgeting will be elaborated on:
1. Percentage of the available time;
2. Pokering the required effort;
3. Pokering the required effort in combination with being 1 sprint behind.

A. Percentage of the available time

This method uses reserving a percentage of the total available test time for test automation as a starting point. A frequently used percentage, distilled from various projects, is

20% of the available test time. Suppose that for 100 hours of testing time 20 hours are used to do test automation. A part of these 20 hours is then used to make the test automation future proof.

By monitoring the actually needed capacity during each sprint, a realistic percentage can be established eventually. The velocity [19] becomes more and more accurate. The question is how reliable such a number is? Does a fixed number allow you to automatize everything that has to be automatized?

The risk is that in, for example, a sprint, not everything can be tested automatically, since the amount of work requires more time than can be realized in the time that is available. A debt is build up which either has to be removed during a next sprint, or in order to finish the amount of work scaling up is done. One of Agile's features is a shared team effort. Developers can support in test automation but this will be at the expense of other work, which puts pressure on the velocity and leads to not being able to realize all the selected product backlog items.

The way of budgeting, as described here, is a very basic way of budgeting, which begs the question of how much functionality can be tested automatically given the framework conditions. The advantage is that you always know how much time is available for test automation.

B. *Pokering the required effort*

Another way of budgeting is applying poker planning [20] specific for test automation. Perhaps initially this might seem like reserving a percentage of time. Initially, pokering the effort uses the brain power of the entire team to reach an actual estimation of the required time.

By placing the items which qualify for test automation on the product backlog, insight will be given into the amount of test work which has to be automatized. Pokering items also provides insight into whether the amount of work fits the current sprint. If the necessary effort is large one can decide to develop less functionality, so that developers can assist in developing the necessary test automation.

This way of budgeting has some caveats to take into consideration. Is the to be test automatized item on the backlog of sufficient depth to determine the scope properly? The second observation has to do with the stability of the features for which test automation has to applied. Is the team only capable of automatizing the test on unit level or also all the features itself?

C. *Pokering the required effort in combination with being 1 sprint behind*

To obtain a larger predictability of the work that has to be done, you can choose to start the test automation in the next sprint using the version of software which was produced in the previous sprint.

This approach has a number of advantages. The software which qualifies for test automation has reached a level of stability which makes it suitable for test automation. Another major benefit is that more detailed information is available with regard to functionality. After all, software has already been produced, which makes it easier to determine which effort is necessary to automatize the tests. Counting the number of functions goes back to the method of budgeting as applied within the TestFrame methodology [17].

This way of budgeting overlooks an important Agile principle namely the fact that working software has to be produced at all times. As a team you cannot guarantee that all software from for example a sprint works, simply because you can no longer test everything manually.

VI. A DEVELOPED EXAMPLE

To give an idea of the various elements influence on the needed capacity, a fictive example has been developed.

BASIC REQUIREMENTS:

Activity	Required capacity in hours	Calculating factor
Testing	1000	
Way of budgeting: 1 (fixed percentage)	200	
Future proof: Reusable: H Repeatable: H Transferrable: L		 1.2 1.2 0.8
Scope test automation: chain test		1.5
Relevant general elements Additional training: License costs		4 persons 1 day €1000, -- per day 4 licenses €1000, -- per year
Hourly fee		€75

TOTAL AMOUNT:

Activity	Calculation	Amount
Required capacity	(200 x 1.2 x 1.2 x 0.8) x 1.5 x €75	€25.920
Training costs	((4 x 8) x €75) + 1000	€3.400
License costs	4 x €1000	€4000
Total costs:		€33.320

VII. COLLECTION OF THE DATA

In the previous sections a few methods are described to estimate test automation in an Agile context. Till now there is no real evidence which method is the best. A first attempt was made as described in Section II. To verify the described estimation method a new survey will be set up to collect the data based on Table V.

TABLE V: COLLECTION OF THE DATA

Project	Type of Initial cost	Prio	Continuity costs	SDLC	Estimation method	Estimated hours	Actual hours

The major problem with the first survey was the timeframe. It was to short to collect data from different customers and projects. The new survey will take place during a period of two years to collect the required data as a base for a proper analysis. At least the data of 100 projects will be collected.

VIII. RETURN ON INVESTMENT

Initially, test automation costs money. As indicated, various elements have to be put in place before test automation can really be applied. The question is when the required investment will be recouped. Tied to this question is the question: what you will earn exactly? Soon thoughts will go to quantitative aspects. However, when it comes to return on investment (ROI) qualitative aspects also play a part. Table VI describes a number of aspects that show how you can recoup the investment.

TABLE VI; ASPECTS RELEVANT FOR THE ROI

Aspect	Description
Shortening test execution time [21]	Manual execution has been replaced by test tools by means of which test automation can be executed in so called off-peak hours. Besides this, a test tool is many times faster than a human being.
Prevention of regression	Because of the acceleration in test execution it has become easier to execute all automatized test scripts. Insight into possible regression can be gotten quickly.
Impact analysis in case of modifications	By executing automatized test scripts in the first sprint, insight into the suggested modifications can be gotten quickly. This can be especially beneficial in a Devops environment.
Time to market	By raising the test execution power, the company can enter the market much faster than its competitors.
Independence	By automatizing the functionality, a company becomes less dependent on a few functional experts. This expertise can be used in other parts or parts of which automation is not useful.
Reliability in the execution	The execution of the automatized test always happens in the exact same way. This provided insight into the stability of the software.
Uniform way of reporting	The test tool generates reports. These describe in detail what happened during the test execution. This makes it easier to track faults and takes less time.
Quality to market	The accuracy of tests gives a good insight into the quality and stability of the information system.

IX. CONCLUSIONS AND FUTURE WORK

This article describes three ways of thinking on how to budget future proof test automation in an Agile environment. These ways of thinking came into being because the existing methodology did not support a reliable budget sufficiently. They will have to be tried and tested in practice by means of a case study. Data has to be collected in order to eventually develop a balanced way of budgeting. Lastly, the article describes the benefits of applying test automation in an organization.

ACKNOWLEDGMENT

I would like to thank Marcel Mersie and Danny Greehorst for their contribution and giving me some of their precious time. Furthermore, I would like to thank the companies where I could set up test automation and develop the in this article mentioned three ways of thinking.

REFERENCES

[1] J. van Rooyen, “Effort estimation test automation in an Agile environment,” Valid2016, 2016.

[2] E. van Veenendaal, ”The Testing Practitioner, hoofdstuk 1,” UTN, 2002.

[3] E. van Veenendaal, M. Posthuma ”Testwoordenboek, blz. 112,” UTN, mei 2010.

[4] M. Fewster, “Common mistakes in Test Automation,“ https://www.stickyminds.com, 2001, 2018.06.15.

[5] Agile manifesto: Agile manifesto, [online]. Available from: www.agilemanifesto.org, 2018.06.15.

[6] S. Hoogendoorn, “Dit is Agile,” Pearson, 2012.

[7] M. Fowler: Continuous Integration, [online]. Available from: https://martinfowler.com/articles/continuousIntegration, 2018-06-15.

[8] Blazemater: The advantages of Manual vs Automated Testing, [online]. Available from: https://www.blazemeter.com, 2018.06.15.

[9] B. vd Burgt, I. Pinkster, “Succesvol testmanagement, een integrale aanpak, blz 110-111,” tenHagenStam, 2003.

[10] D. Greefhorst, M. Mersie, J. van Rooyen, “Principes van testautomatisering,” Computable, 2015.

[11] DJ de Groot, “Testgoal,” SDU, 2008.

[12] J. vd Laar, “Technieken voor plannen en begroten van test projecten,” Testnet voorjaarsevent, 2009.

[13] M. Karlesky, M vd Voord, “Agile projectmanagement,” Embedded systems conference Boston, 2008.

[14] ISBSG, “ISBSG database,”

[15] Guru99: Automation Testing for Agile methodology, [online]. Available from: https://www.guru99.com, 2018.06.15.

[16] European Commission: GDPR, [online]. Available from: https://ec.europa.eu/info/law/law-topic/data-protection, 2018.06.15.

[17] C. Schotanus et al, “Testframe, hoofdstuk 6,7,8,” Academic Service, 2008.

[18] M. Siteur, “Automate your testing, sleep while you are working, blz 143-144,” Academic Service, 2005.

[19] Improvement: Term velocity, [online]. Available from: www.improvement-services.nl, 2018.06.15.

[20] Improvement: Term pokerplanning, [online]. Available from: www.improvement-services.nl, 2018.06.15.

[21] Test Automation: Test Automation Snake Oil, [online]. Available from: www.satisfice.com, 2018.06.15.