APPLYING SURVEYS AND INTERVIEWS IN SOFTWARE TEST TOOL EVALUATION

Päivi Raulamo-Jurvanen
ITEE, M3S
University of Oulu
Oulu, Finland
paivi.raulamo-jurvanen@oulu.fi

Simo Hosio
ITEE, UBICOMP
University of Oulu
Oulu, Finland
simo.hosio@oulu.fi

Mika V. Mäntylä
ITEE, M3S
University of Oulu
Oulu, Finland
mika.mantyla@oulu.fi

The final authenticated version is available online at https://doi.org/10.1007/978-3-030-35333-9_2.

ABSTRACT

Despite the multitude of available software testing tools, literature lists lack of right tools and costs as problems for adopting a tool. We conducted a case study to analyze how a group of practitioners, familiar with Robot Framework (an open source, generic test automation framework), evaluate the tool. We based the case and the unit of analysis on our academia-industry relations, i.e., availability. We conducted a survey (n=68) and interviews (n=6) using convenience sampling to develop a comprehensive view of the phenomena. The study reveals the importance of understanding the interconnection of different criteria and the potency of the context on those. Our results show that unconfirmed or unfocused opinions about criteria, e.g., about *Costs* or *Programming Skills*, can lead to misinterpretations or hamper strategic decisions if overlooking required technical competence. We conclude surveys can serve as a useful instrument for collecting empirical knowledge about tool evaluation, but experiential reasoning collected with a complementary method is required to develop into comprehensive understanding about it.

***K*eywords** Test automation · Software testing tool · Tool support · Tool evaluation · Case study · Survey · Interviewing.

1 Introduction

Testing and test automation are expected to have potential combining quality with speed and reducing costs. Nevertheless, those tasks are reported to be under-exploited activities in Quality Assurance (QA) [2]. It seems rather easy to search for types of software testing tools, but practically hard to evaluate and select the most suitable one from the plethora of tools. Despite the volume of software testing tools available, practitioners tend to find lack of right tools as an obstacle [18, 2]. Marketing material or promotional tool comparisons tend to focus on desirable benefits, but seem to fail in providing realistic details about prerequisites or related challenges. In software engineering (SE), practitioners tend to find beliefs of their peers more credible than empirical evidence [19, 15].

In this paper, we report a case study [23, 29], a common case of a tool evaluation in the context of software testing. We find it relevant to ask, whether expert advice is accurate and appropriate for tool selection. Case studies are suitable to settings where *how* and *why* questions are favorable, researchers do not have control over variables and the focus is on some contemporary events [16, 29]. The use of multiple sources of evidence is a major strength of case study data collection [29]. We study the different criteria of the tool and its potential, as evaluated by software practitioners in the field, in the form of a survey. To provide a broader view on the concept, we complement the survey with interviews and assess quantitative results of the survey in the light of qualitative data from the interviews. We formulated the following research questions:

- RQ1. *How do practitioners ground their tool evaluations?*
- RQ2. *How to identify possible false expectations from tool surveys?*

To answer our research questions, we will compare the results of both methods for supportive and conflicting perceptions. By triangulation, we intend to capture rich dimensions on the characteristics of the tool [23].

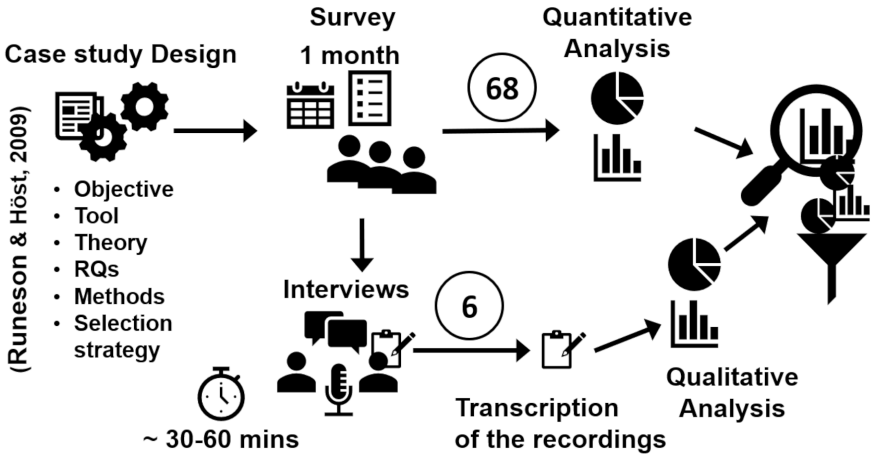

Figure 1: Case study design, complementing a survey with interviews

2 Related Work

Evaluating software testing techniques and tools is time-consuming, expensive and difficult [17, 28]. According to Fenton et al. [4], a single tool evaluation trial, even with a realistic project having realistic subjects, is not adequate, and claims by analytical advocacy are considered insupportable. In academia, publication bias of positive research results may be a problem, especially in stronger sources of evidence [18]. Dybå et al. [3] promoted evidence-based SE (EBSE) as a mechanism to aid adoption of technology related decisions. The research should seek evidence of realization of expected outcomes, potential side effects and causes of those, that can be integrated from both research and practical experience [3]. Sjøberg et al. [25] consider the viewpoint of practitioners as means to explore, describe, predict and explain phenomena.

Murphy-Hill et al. [13] focused on events where a need for a tool arises on its discovery. They reported tool encountering to be the most frequent discovery mode [13]. A widely used tool is likely found useful upon tool discovery [13]. In software projects, the need for a software testing tool is often perceived, but it is problematic to discover and select the most suitable tool(s). Comprehensive understanding of usage habits of software practitioners in a community is seen more reliable than an opinion of just one individual [13]. It is important to understand the experiences, both positive and negative, related to those habits.

Practitioners seem to have common but not systematically applied consensus about important criteria for selecting software testing tools [20, 21]. For example, costs, in general is one frequently mentioned factor for the adoption and use of software testing tools [14, 27, 5, 20, 1]. Cost is an important factor, but not considered to be a characteristic of product quality[1]. Rather, costs are categorized as a tool external factor [21]. In our prior research on tool evaluations [22], we found that practitioner evaluations for a tool, in a survey, may be dispersed. To improve understanding and robustness of the results, we analyze the topic using a complementary method.

3 Case Study Design

We apply a case study as an empirical research method for studying the evaluation of the selected software testing tool, in the context of shared open source software (OSS) ecosystem. For a case study, both the case and the unit of analysis should be selected intentionally [23]. We based the tool selection on our existing academia-industry relations, i.e., availability [23]. The case tool is Robot Framework, an open source (OS), "*generic test automation framework for acceptance testing and acceptance test-driven development (ATDD)*"[2]. The tool is utilized by a set of collaborating companies in our research project, EUREKA ITEA3 TESTOMAT [3]. We could reach practitioners familiar with the tool via the companies, representing an example case of shared OSS ecosystem. See Fig. 1 for the design of the case study.

The primary objective of our study is of exploratory nature. Perry et al. [16] find case studies to be useful for exploratory studies that "*attempt to understand and explain phenomenon or construct a theory*". Kitcheham et al. [8] emphasize

[1] http://iso25000.com/index.php/en/iso-25000-standards/iso-25010

[2] https://robotframework.org/

[3] https://itea3.org/project/testomatproject.html

that although case studies are not scientifically as rigor as formal experiments, those are useful in judging whether some technologies will be of advantage in a setting. To evaluate software testing tools, we need collective information, knowledge from people who have invested in choosing and using tools [20]. For our study, we considered two methods for collecting data, a survey and interviews. Kitchenham et al. [9] outline a survey as "*comprehensive system for collecting information to describe, compare or explain knowledge, attitudes and behavior*". Complementing a survey with interviews allows us to increase the amount and diversity of the data, to develop a more comprehensive understanding about the phenomena and possibly to confirm the validity of conclusions [24, 11, 23].

3.1 Tool Evaluation Survey

The questionnaire[4] included 15 questions. The questions are based on our prior work [20] and on the ISO/IEC 25010 quality model. The approach for the survey tool was adopted from the studies of Hosio et al. [7] and Goncalves et al. [6]. The survey tool was validated by the authors and an industry partner. As a result, we added the options to indicate the basis of the answers (i.e., hands-on experience or generic opinion) and any experience in the development of the tool to the questionnaire. The survey was sent to seven software professionals collaborating in the research project (March 1st, 2018). We requested them to distribute the survey to their colleagues experienced with Robot Framework. To reach users of the tool, the survey was also promoted in Robot Framework Slack and in Twitter (with hashtag *robotframework*). The survey was open for a month.

Background information was given in 80 unique responses. We excluded the responses known to be for testing purposes, and those having only default values. 68 respondents completed the survey (998 unique questionnaire answers for the 15 criteria, in total). The response rate among those having started the survey was 85%. We could not calculate the overall response rate as the number of practitioners having received the link was not known. As the survey was anonymous, we could not ask for reasons not completing or not responding the survey. We used MS Excel and R/RStudio for analyzing the data. We analyzed the numeric data using descriptive statistics and graphical visualization (boxplots).

3.2 Interviews

While the purpose of the survey was to collect quantitative data, the interviews were designed to collect rich descriptions, i.e., qualitative data. The interview questions were based on the questionnaire, to have the experts elaborate on their personal experiences. We recruited six volunteering practitioners, using convenience sampling [10] for interviews, from our contacts via the TESTOMAT project and Robot Framework Slack.

The objective of the interviews was descriptive and explanatory. The interviews were semi-structured, the questions open and the content and order of the questions the same for all interviewees although the questions could be answered freely. The interviews were conducted via Skype (March-April 2018). To mitigate the risks of misunderstandings and loss of information, we requested each interviewee the permission to record the interview. To minimize the bias related to different interviewers, the interviews were conducted by one of the authors. The recordings were analyzed in NVivo 11. The data were coded against the survey criteria, to find explanations in the descriptions, i.e., to *"illuminate the quantitative findings"* [24].

4 Results

First, we present the demographics of the survey respondents and interviewees in Section 4.1. Thereafter, we present the overview of the results from both the survey and the interviews in Section 4.2. To build a comprehensive picture of the phenomena under study, we will triangulate the results from both methods, in detail in Section 4.3 and answer our research questions in Section 5.

4.1 Background Information

Unsurprisingly, most of the survey respondents (54%) work in Finland (the tool originated in Finland, and the survey was initially promoted via Finnish collaboration companies). Most questionnaire answers (97%) were based on hands-on experience using the tool. Of the respondents, nearly 6% had contributed to the development of *either* the core of the tool *or* both the core of the tool and related libraries, about 21% to the development of related libraries, while majority (63%) had not contributed at all. About 85% of the respondents had used Python in their work and 50% Java. Six respondents had not used either of those while two had not reported (or used) any programming languages.

[4] https://drive.google.com/open?id=1xzXG5ypANvOCbMdRyAyUnmYd0N24VCr4

Table 1: Experience in years

Source	Experience in	Min	Max	Mean	Median	Mode
Survey (68)	Industry	3.0	33.0	12.4	10.0	5.0
	Current Role	0.0	14.0	3.5	2.0	1.0
Interviews (6)	Industry	11.0	18.0	14.2	13.5	11.0
	Current Role	0.0	6.0	2.2	1.5	0.0

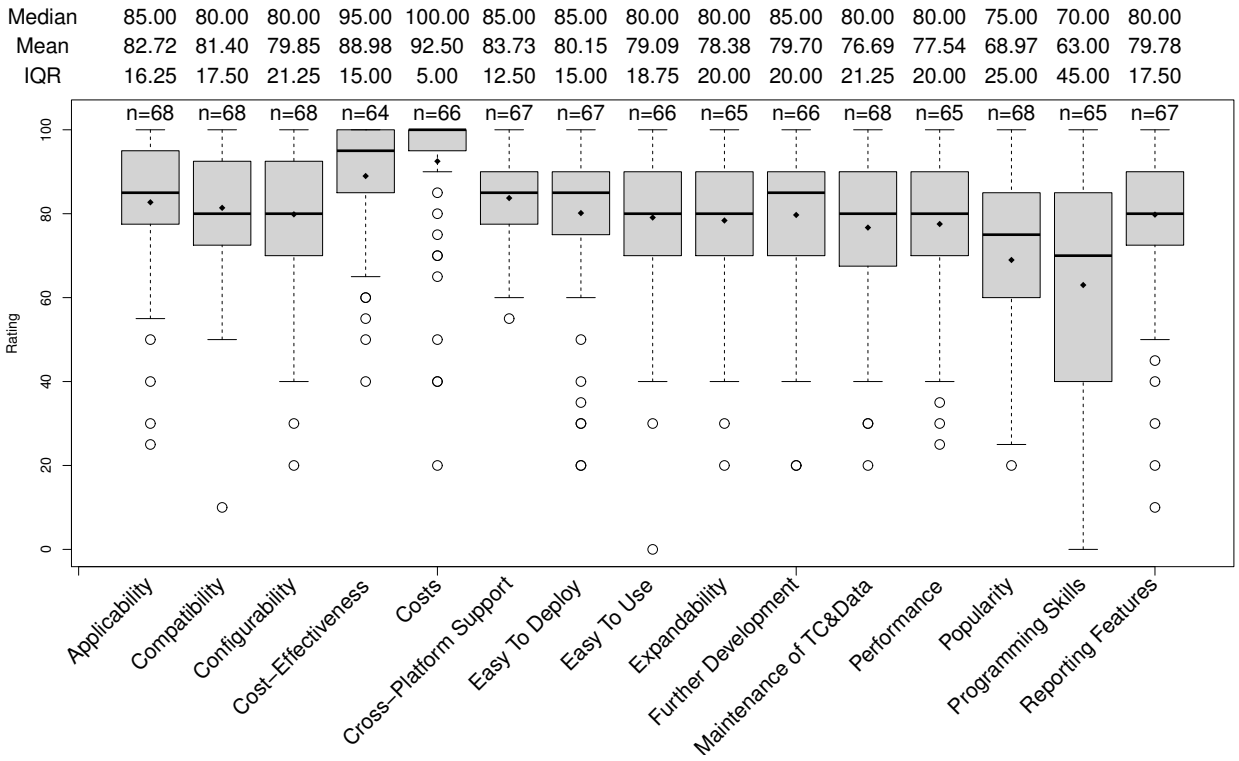

Figure 2: Variability of questionnaire answers for the criteria in the survey

The interviewees represented different companies, in different domains (from consulting to cyber security). Regarding the experience *in the industry*, the interviewees were more experienced than the survey respondents in areas other than the maximum number of years, see Table 1. Three interviewees had not contributed to the development of the tool, two had contributed to the development of libraries, and one to both the core of the tool and libraries. Five interviewees had been using Python and two Java in their work while one had not been using either of those.

4.2 Overview of Data from Tool Surveys and Interviews

In the boxplots[5] of questionnaire answers, see Fig. 2, the median is shown as a horizontal line and the arithmetic mean as a dot. The interquartile range (IQR) describes the middle 50% of the data. By default, in R, the formula for calculating the upper whisker is $min(max(x), Q3 + 1.5 * IQR)$, and for the lower whisker $max(min(x), Q1 - 1.5 * IQR)$.

The interview transcripts were coded according to the criteria (related sentences were coded accordingly) as the questions were based on those. We classified the thoughts for the criteria from the interviews as positive, negative and neutral, based on the sentiment, see Table 2. The column "*Rank*" in Table 2 indicates the order of the column "*n*", the total number of coded items for each criterion. The columns "*Positive*", "*Negative*" and "*Neutral*" items include the number of coded items and the number of associated interviewees. Most of the coded items were either positive or neutral (roughly about 40% both), see example statements in Table 3. The quantity of coded items provides an overview of the topics of interest or familiarity, those the interviewees felt easy, comfortable or important to discuss. A high positive rank may indicate favorable attitude towards the criterion while a high negative rank may reveal concerns or problems. Next, we will discuss the criteria in the light of the thoughts from the interviews.

[5]https://www.rdocumentation.org/packages/graphics/versions/3.5.3/topics/boxplot

Table 2: Interview data coded as Positive, Negative & Neutral items

Rank	n	Criterion	Positive Items		Negative Items		Neutral Items	
			#	Interviewees	#	Interviewees	#	Interviewees
4	33	Applicability	16	6	7	3	10	4
9	26	Compatibility	15	6	1	1	10	4
15	17	Configurability	11	6	0	0	6	4
10	25	Cost-Effectiveness	13	4	2	2	10	5
10	25	Costs	4	3	4	3	17	5
13	18	Cross-Platform Support	8	4	0	0	10	5
12	20	Easy To Deploy	8	5	11	6	1	1
2	44	Easy To Use	18	6	7	2	19	5
3	36	Expandability	23	6	3	2	10	3
1	52	Further Development	16	5	12	4	24	4
6	31	Maintenance	7	3	11	5	13	6
13	18	Performance	4	4	7	5	7	3
6	31	Popularity	11	5	6	4	14	6
5	32	Programming Skills	4	3	10	4	18	6
8	28	Reporting Features	17	4	3	2	8	4
Total	436		175		84		177	

= Number of coded items, Interviewees = Number of unique respondents

Table 3: Example statements from the interviews

Criterion	ID	Type	Statement
Applicab.	2	-	"Many times people try to use Robot Framework for many other purposes for which it is not the best tool."
	1	+/-	"It is good for mature cases, where the lifespan of the product or system is long."
Compatib.	4	+	"It is very compatible. The community has created a large number of libraries for integrating with other tools."
	2	-	"In my opinion the problem is that people expect it to work out-of-the box and that is not how it always work."
Configurab.	3	+	"It is an advantage that we can easily do configurations from files and we can avoid hard-coded items."
	4	+	"I think it is highly configurable, meaning there are several ways in which to override settings, several ways in which to specify settings."
Cost-Effect.	3	+	"Cost-effectiveness and re-use are developed along with the experience and know-how of the tool and its usage."
	6	-	"It (test automation) may not even be any cheaper than manual work."
Costs	4	+	"They are, I mean the only cost is training, so they are very low."
	6	-	"Deployment costs, a lot, to gain benefits."
Cross-Pl.S.	1	+	"You can run the tool basically for any platform where needed."
	4	+/-	"Depending on the environment you are running in, you can specify the configuration for that, at run-time."
Easy to Dep.	2	-	"You don't have to have in-depth know-how but for a non-technical person deployment may be quite difficult."
	1	+/-	"Building the test environment is many times the biggest challenge in every project. But it not necessarily due to the selected tool but more about the characteristics of the underlying system."
Easy to Use	2	-	"To use the tool efficiently, you need to have technical competence. That is not always clear."
	6	+/-	"There are many libraries, developed in many ways for different purposes and it's more of a question which libraries you use and how easy that is."
Expandab.	5	-	"For developing libraries you will need programming skills and more understanding about the system."
	6	+/-	"It is open source and if you want to touch the core system, that is doable. And the libraries, those are, as a general rule, open source and you may expand those, too."
Further Dev.	3	+	"There is the Foundation developing the tool and it convinces again, end customers or other customers, that there is sustainable development for the tool."
	5	-	"It was some web-automation demo I found, and I tried to use it... it did not quite work and the instructions would require small elaboration and the links updating... then it would be easier for the people to get started."
Maintainab.	1	+	"Another good feature is the tool is keyword-based, at the core of the maintainability, how you can create layers of keywords and how you can create meaningful abstractions for the test cases."
	4	+/-	"The way in which you write your tests and your keywords... will essentially determine how easy or how difficult it is, for maintenance and re-use in the future."
Performance	2	+	"The tool itself is good, or has always been adequate, so I have not had such problems."
	1	+/-	"At the end of the day, it is up to the user of the tool."
Popularity	6	+	"In Finland, the tool is well-known and you would have to have good reasons for selecting another tool."
	3	-	"A community-type tool like this has not had credibility in all branches of industry."
Progr. Skills	5	+	"I have heard that people without any programming skills can create test-automation scripts with the tool, and that is based on the keywords."
	6	-	"If you have a misconception about Robot Framework, library or any other framework, that you do not have to write any code, that is a terrible misconception."
Reporting F.	4	+	"As long as the tests were written well, anybody without programming skills should be able to read the log and understand what happened."
	5	-	"There is not much visualization. If you want something more, you have to build it yourself, that is my opinion."

ID=Interviewee ID, Type: (+)=Positive, (-)=Negative & (+/-)=Neutral

4.3 Analysis of the Criteria

4.3.1 Applicability

The participants evaluated the applicability of the tool to their tasks, methods and processes. The interviewees highlighted that the tool is applicable for various different purposes, provided the users have relevant technical skills. However, applicability was seen as a dilemma. A multitude of possible contexts and ways of utilizing the tool prevents generalizing, e.g., providing detailed guidelines and best practices. "*The fact that the tool is desinged to be a command line tool enables its use in many contexts, in many operating systems. So, I would say rather well.*"(#2)[6]. "*In terms of tools, it is very applicable and then, in terms of the language, the structure of writing tests in Robot Framework, that is also very applicable, because the group that I am working in, this is mostly manual testers.*"(#4).

4.3.2 Compatibility

Compatibility of Robot Framework with the existing tools was not considered to be a problem threshold by the interviewees. In the survey, it was the only criterion for which the arithmetic mean was greater than median. The interviewees pointed out that there are always issues that could be simply improved, in general. For example, there were needs to have the tool started via REST-API, or to integrate it with other tools. "*And then, whenever there is not a particular library for integrating with a tool, it is often to make a tool available via an API, so that Robot simply then just calls the tool's API.*"(#4).

4.3.3 Configurability

Our participants evaluated the possibilities for configuring the tool for their needs. The interviewees found configuration of both the run time environment and reporting for a test set very practical. None of the interviewees came up with negative coded items, but *Configurability* had the least coded items. Thus, it could be seen as an issue having no use for emphasis. For those having experience with configurations, the tool seems configurable. "*There are moderately a lot of different options to configure the handlig of the tests, the kind of tests you want, how you want to view the report, format and all that.*"(#6).

4.3.4 Cost-Effectiveness

Surprisingly, there were not very many coded items, and most of those were positive or neutral. A tool, as test automation in general, is cost-effective if applied the right way, at the right time alongside the development work. What is the right way and the right time are volatile and contexts specific concepts. Amount of money was seen as a likely issue for consultants and clients to discuss, but difficult to verify in real life. One of the interviewees pointed out cost-effectiveness as a way to prioritize the work load. They emphasized the fact that test automation helps to become faster, i.e., in the best case it helps to deliver software more often, in smaller batches and with better quality. "*If one starts from the scratch, it takes some time and some studying. Like everything else, from scratch, so I do not see that as a problem .*"(#3).

4.3.5 Costs

The tool was evaluated for expected costs (for acquisition and use). In survey responses, the median was the highest. We expect the main reason for that to be the fact the tool is free. *Costs* was the criterion with the most outliers (given by 10 respondents having been in the software industry on average 12.9 years). In the interviews, there were 25 coded items, most of which were neutral (17). The interviewees highlighted the importance of understanding the inevitable costs related to the tool. The required resources for setting up and maintaining a system (e.g., people, training or time) depend on the context, and have direct effect on *Costs* and *Cost-Effectiveness*. "*The problem is, how you organize the use in the company... where the costs come from, that holds true how much you have available resources, people, how technically competent they are.*"(#2). "*And its [test automation] maintenance costs, a lot.*"(#6).

4.3.6 Cross-Platform Support

The participants evaluated their view on cross-platoform support of the tool. In the interviews, there were no negative coded items. The tool was considered to have good support for different platforms. "*The support for SUTs comes via the libraries and the support is broad. We have tested all kinds of systems, from elevators to insurance systems... and network protocols, and dynamic web-applications, so it is very, very versatile, in that sense.*"(#1). "*You may*

[6]#n = ID of the interviewee

run Robot anywhere where you can run Python, which means from mainframes to Raspberry Pi's and small micro controllers."(#6).

4.3.7 Easy to Deploy

The participants considered the initial efforts to take the tool into use. In the interviews, there were the 2nd most negative items (of all criteria). All interviewees had stated one or more negatively coded items. Although the interviewees found the deployment to be rather easy, they emphasized the need for technical know-how, preferably with Python. The interviewees highlighted the fact that the difficulties may not only be related the tool but also to test automation, in general, and to the underlying system itself as well. "*When considering the easiness of the deployment, it is difficult to disassociate the tool, the system and all that is around it.*"(#1). "*In a way, it is easy to deploy, but the difficulty lies in that it truly requires planning.*"(#3).

4.3.8 Easy to Use

We requested the participants to consider their perceptions of *Easy to Use*. In the interviews, the criterion had the 2nd most coded items. Those were mainly positive (18) and neutral (19). Thus, we assume interviewees felt easy to talk about their experiences using the tool. The interviewees pointed out the need for technical know-how. They thought that the concept of test automation in general, in the given context, may be difficult to comprehend. Possible wrong choices or mistakes in the setup may require unexpected changes to the test sets (or even to the system) later on. Effective use of the tool requires careful planning. "*Planning must be done the right way, meaning, that you can also make bad choices that may backfire on you later.*"(#3). "*The people on my teams have really seen the effectiveness of it, and have enjoyed working in it.*"(#4). "*Writing the actual tests is easy and clear, of course.*"(#5).

4.3.9 Expandability

For *Expandability*, the participants could share their views on the possibility to remold or expand the tool. An OSS tool has its benefits and its downsides when considering expandability. There may be an active community of software practitioners developing the features of the tool. Nevertheless, one needs programming skills and understanding of the problem area and/or the architecture to make changes. "*With Robot Framework, you need to be careful whether to talk about the tool itself or the ecosystem, as many issues that have been discussed over the years, that would be good to have in a tool in one way or another, are such that could already be done as an extension (library) and in that sense there is necessarily no need to modify the tool itself.*"(#2). "*There are true programming languages to use, and the sky is the limit, so, expandability can be achieved with those.*"(#1).

4.3.10 Further Development

The participants could evaluate whether they find the *Further Development* of the tool (by the OS community) active or not. The criterion was the most discussed among the interviewees and the coded items were mainly neutral (24). What could not be understood from the survey was the dualistic nature of the tool. The tool consists of two fundamental entities, the core tool and related ecosystem (libraries and tooling type of development). The core tool is a framework using the functionality provided by the ecosystem. The two concepts are distinct, developed and maintained separately. The core tool itself is rather stable. It is the ecosystem that needs to change according to the conditions around the tool, in the industry, in general. The interviewees emphasized that the core tool is well designed for adding new functionality via libraries. The documentation must be up-to-date to provide value to the users.

The Robot Framework Foundation supports the resourcing for the development of the core tool. "*We have this foundation, which will support development, collect membership fees from member companies to finance basic updates to Robot [Framework] to keep it compatible and to work in all platforms in the future, too. And of course, there is the open source community that contributes a lot to the libraries.*"(#1). "*The fact that the foundation supports the development, it is a good thing.*"(#3). It was noted that if the difference between the core tool and the ecosystem is not understood, a low quality library may invite unfounded criticism for the core tool. "*What I hope is that the discussions, in general, would move from the core tool to the libraries and testing... and there would be the common understanding that if something goes wrong, it is not necessarily Robot Framework but some library, instead. And even though Robot Framework has a public site for reporting bugs, many of those are closed because those are not related to the core tool but to some specific library.*"(#2). *"You should develop all libraries and other type of development following the good software development practices, but what those really are, that is a good question.*"(#2).

4.3.11 Maintenance of Test Cases & Data

The participants could evaluate maintenance and re-use of test cases and data. In the interviews, the coded items were mainly neutral (13) and negative (11). It became clear the practitioners find maintenance of test cases and data laborious and costly, if not planned carefully. Furthermore, practitioners maintaining the test system must need competence for the tasks. The help of possible external consultants must meet the needs and competence level of the clients. "*I think that you have to be very careful in setting up your library of tests, and it can be very simple to create a maintenance headache for yourself.*"(#4). "*Development of libraries may be challenging, development of keywords, what I have heard, may easily explode.*"(#5).

4.3.12 Performance

We queried about the *Performance* of the tool for its purpose. In the survey, based on the boxplots, *Performance* was evaluated as many other criteria. In the interviews, there were not many coded items. The interviewees considered the tool itself to be fine, performance-wise, although they found performance to be a difficult concept to measure. The problems with performance can be related to the system under test (SUT), set up of the overall test system and its users, not just the tool. So, this is not a self-contained criterion. "*So, as you are using the tool, the tool itself performs just fine, but... there are things like parsing files, for example, that is probably done faster outside of the tool.*"(#4).

4.3.13 Popularity

In the survey, *Popularity* had the 2nd lowest arithmetic mean. The interviewees noted that the tool is rather well-known in Finland and in the Nordic countries, but not globally. According to the interviewees, practitioners seem to rely on positive hearsay and meet-ups, as well as testimonials from reference companies. Companies may be reluctant to change an invested tool, even if the tool was not found as the best option in the task. "*Testing as a field suffers a bit from the fact that information is not shared the similar way as in software development.* "(#2). "*I'm not actively involved in the community but I have been following the Slack channel, the Slack work space, which I think is great... I think the most important enablers for future development are the community itself, the fact that the community is welcoming, that the community is helpful.*"(#4).

4.3.14 Programming Skills

The participants assessed the level of required *Programming Skills*. In the survey, the criterion had the lowest arithmetic mean. In the interviews, it was the 5th most coded criterion. Programming skills and technical skills are issues of importance for the use of the tool. A high variance in questionnaire answers and a negative nuance in the coded items from the interviews, suggests that technical competence, in general, is of importance. Building and maintenance of the test environment, and development of needed functionalities are linked to performance and cost-effectiveness of the tool yet tool cost itself is not the issue. "*Would be good to understand the basic concepts of programming for creating test cases in the right abstraction level, which impacts maintainability.*"(#1). At the time of the study, the testing capabilities of the tool could be extended by test libraries implemented with Python or Java[7]. So, it is not only about programming skills, but also about specific programming languages.

4.3.15 Reporting Features

The participants assessed the set of reporting features in the tool to be limited or rich. In the interviews, the coded items in the interviews were mainly positive (17). The interviewees emphasized *Reporting Features* as a tool not only for the developers, testers and managers but also for the clients. The tool provides logs for finding bugs and understanding the behaviour of the system, and rich data for visualization. Programming skills are not needed for reading the logs or reports, but for creating rich reports with charts and graphs (for connecting external tools). "*An example of tasks where you don't need programming skills, I would say, reading the logs and reading the reports.*"(#4).

5 Discussion

Tool evaluations depend on the interpretation of a construct under study, i.e., have a degree of subjectivity [26] but also validity as measures [12]. The questionnaire answers are results from plain realism acquired from personal experiential knowledge for reasoning about each criterion as such. We conducted interviews to grasp detailed understanding about the findings.

[7] https://robotframework.org/

For ***RQ1.*** *"How do practitioners ground their tool evaluations?"* we assessed the foundation for the responses of the interviews. The interviewees reflected on their insights with rich, informative examples from real life, verbalizing their reasoning. One emphasized testable requirements and realistic benefits for the test system to be built. Another noted that test automation is expensive in the short term, but may be very economical, in the long run. Importantly, test automation is efficient and prudent use of resources in the development process. "*It (test automation) helps you to be faster... helps you to achieve the goals and to release faster, more often, in small batches. It helps you to achieve better quality, if you have done it the right way.*"(#6).

The different criteria are highly interconnected. The interviewees connected criteria like *Costs*, *Cost-Effectiveness* and *Expandability* not only to the to technical competence, but also to the level of *Programming skills*. Evaluation of a tool criteria may be related to the level of knowledge of the system, in general. "*Building the test environment is the biggest challenge, in general, in every project.*"(#1). "*Sometimes, it is really difficult to find the right way to apply your solution... efficient use of the tool requires some level of technical competence.*"(#2). "*Test automation is always a programming issue, and if you want to have test automation, you need to be able to program.*"(#6).

The issues regarding *Costs* and *Programming skills* are interconnected to the main characteristics of OSS[8]: free to use and source code accessible to all. An OS tool is free and expandable, and there may be an active OS community developing it. Yet, tool related tasks require investments (e.g., competent people, time and money), within contexts of the organizations utilizing the tool and the community developing the tool. Lack of technical competence or programming skills seem disadvantageous for tool usage and evaluations. Well-argued experiences from expert practitioners allow to reveal unexpected problems, clarify common misconceptions or confirm understanding about tool criteria. Neither single criterion nor grounded reasoning by a peer should be decisive. "*It is the accumulation of that information, not the ratings themselves, that is decisive* [12]".

With ***RQ2.*** *"How to identify possible false expectations from tool surveys?"*, we focused on finding possible potentially misleading or restrictive perceptions. From the boxplots (see Fig. 2), we could observe many of the criteria, for example, *Easy to Use*, *Expandability* and *Performance*, to be of similar shape, and to have both the median and the length of the whiskers roughly the same. However, *Costs* and *Programming Skills* were the criteria having the lowest and the highest variance in the questionnaire answers, respectively. Furthermore, the findings for those criteria from the two methods were contradictory.

The respondents agreed the most on *Costs*, majority finding costs for acquisition and usage of the tool to be very low. The finding suggests they considered the licensing fee, not costs of required training or using the tool. Thus, it seems we missed to cover different aspects of *Costs* in the survey. However, it is possible they had not faced costs (as *extra costs* but *work*) or needs for training. "*If you have enough of technical competence, at that stage, costs will be trivially small, because you just re-prioritize the tasks of those people for Robot Framework*"(#2). The interviewees, highlighted that software test automation costs, a lot, no matter the tool. "*Test automation is always a big investment for a company. It costs, always. Costs is not about just getting the software, it is about using it, setting up the infrastructure, learning to use it, creating, maintaining, all that. It includes a lot of costs and Robot Framework is not an exception.*"(#6).

Programming skills had the lowest arithmetic mean and median of the criteria. Role and tasks of a practitioner using the tool impact the level of required programming skills. The interviews revealed the dualistic nature of the tool. The Robot Framework foundation is resourcing the development of the core tool, but has no control over resourcing or quality of the ecosystem around it. "*If some of the libraries does not support your thing, you are basically on your own, you need to build the library yourself.*"(#6).

6 Threats to Validity

As our target population for the use case was very specific, i.e., software practitioners experienced with Robot Framework, we could not rely on random samples. We used non-probabilistic sampling methods: convenience sampling complemented with snowball sampling [10]. We expected to have representative samples of the target population, i.e., software practitioners experienced on using the tool (in their contexts). While the survey respondents (n=68) were expected to be experienced in using Robot Framework, we could not assess their experiences of each criterion. In a survey, the likelihood of participation may be related to negative experiences [10]. To mitigate deficiencies in collecting data and to understand the phenomena better, we used a complementary method, interviewing (n=6), and triangulated the results with those from the survey.

As the participants of the study were mainly from Finland, the results may be biased by confounding factors e.g., knowledge of the tool or contexts. However, the survey participants came from 13 countries, 10 participants from other European countries and 21 from outside Europe. Experience in the development of the tool was seen as in depth view

[8]https://opensource.org/osd

of the tool. The cohesion and consistency of the results from the survey are impacted by the facts that tool evaluations are highly subjective, and we could not control the contexts or the constructs. Thus, our results are not generalizable as such, but provide a snapshot of opinions, in a given time, and are presented to be useful for analyzing dissenting opinions.

7 Conclusions and Future Work

We complemented a survey with interviews for analyzing differing opinions about characteristics of an OSS testing tool. Our survey revealed *Costs* and *Programming Skills* to be quite different from the other criteria. The interviews clarified a tool may be free, but investments carry costs which, in turn, are always context specific. While cost is not a quality characteristic of a tool, tool related costs are restrictive and can hamper strategic decisions. Technical competence is vital for efficient tool adoption and usage, and development of the tool. A tool is no silver bullet but a facility for re-prioritizing tasks in the software development work.

We conclude that complementary methods can dispel common misconceptions about characteristics or usage of a tool, or about software test automation as a whole. Contradictions should merit further studies and reasoning, in the context. There is a need for more, in depth research on software testing tool evaluations. In the future, we plan to study viewpoints of the practitioners, in more detail. Academic research on software test tool criteria can help the practitioners to view the forest from the trees, and focus on achievable goals.

8 Acknowledgments

The work was supported partially by research Grants No.: 3192/31/2017 from Business Finland for the EUREKA ITEA3 TESTOMAT project (16032), and No.: 286386-CPDSS from the Academy of Finland for the CPDSS project.

References

[1] Bhargava, S., Guleria, S., Gaurang, A.: A study on the current trends in software testing tools. International Journal of Advanced Research in Computer Science **8**(5), 129–131 (2017)

[2] Capgemini, Micro Focus and Sogeti: World quality report 2017-2018 (2017), `https://www.sogeti.com/globalassets/global/downloads/testing/wqr-2017-2018/wqr_2017_v9_secure.pdf`, last accessed 5 Jun 2019

[3] Dybå, T., Kitchenham, B.A., , Jørgensen, M.: Evidence-based software engineering for practitioners. IEEE Software **22**(1), 58–65 (2005). https://doi.org/10.1109/MS.2005.6

[4] Fenton, N., Pfleeger, S.L., Glass, R.L.: Science and substance: a challenge to software engineers. IEEE Software **11**(4), 86–95 (1994). https://doi.org/10.1109/52.300094

[5] Garousi, V., Zhi, J.: A survey of software testing practices in canada. Journal of Systems and Software **86**(5), 1354–1376 (2013). https://doi.org/10.1016/j.jss.2012.12.051

[6] Goncalves, J., Hosio, S., Kostakos, V.: Eliciting structured knowledge from situated crowd markets. ACM Trans. Internet Technol. **17**(2), 1–21 (2017). https://doi.org/10.1145/3007900

[7] Hosio, S., Goncalves, J., Anagnostopoulos, T., V.Kostakos: Leveraging wisdom of the crowd for decision support. In: Proceedings of the 30th International BCS Human Computer Interaction. pp. 1–12. BCS Learning & Development Ltd. Swindon, UK (2016). https://doi.org/10.14236/ewic/HCI2016.38

[8] Kitchenham, B., Pickard, L., Pfleeger, S.L.: Case studies for method and tool evaluation. IEEE Software **12**(4), 52–62 (July 1995). https://doi.org/10.1109/52.391832

[9] Kitchenham, B.A., Pfleeger, S.L., Pickard, L.M., Jones, P.W., Hoaglin, D.C., Emam, K.E., Rosenberg, J.: Preliminary guidelines for empirical research in software engineering. IEEE Transactions on Software Engineering **28**(8), 721–734 (2002). https://doi.org/10.1109/TSE.2002.1027796

[10] Kitchenham, B.A., Pfleeger, S.L.: Personal opinion surveys. Guide to Advanced Empirical Software Engineering pp. 63–92 (2008). https://doi.org/10.1007/978-1-84800-044-5_3

[11] Lethbridge, T.C., Sim, S.E., Singer, J.: Studying software engineers: Data collection techniques for software field studies. Empirical Software Engineering **10**(3), 311–341 (Jul 2005). https://doi.org/10.1007/s10664-005-1290-x

[12] Linacre, J.M.: Judge ratings with forced agreement. Transactions of the Rasch Measurement SIG American Educational Research Association **16**(1), 857–858 (2002)

[13] Murphy-Hill, E., Lee, D.Y., Murphy, G.C., McGrenere, J.: How do users discover new tools in software development and beyond? Computer Supported Cooperative Work (CSCW) **24**(5), 389–422 (Oct 2015). https://doi.org/10.1007/s10606-015-9230-9

[14] Ng, S.P., Murnane, T., Reed, K., Grant, D., Chen, T.Y.: A preliminary survey on software testing practices in australia. In: Proceedings of the 2004 Australian Software Engineering Conference. pp. 116–125. IEEE, NJ, USA (2004). https://doi.org/10.1109/ASWEC.2004.1290464

[15] Pano, A., Graziotin, D., Abrahamsson, P.: Factors and actors leading to the adoption of a javascript framework. Empirical Softw. Engg. **23**(6), 3503–3534 (Dec 2018). https://doi.org/10.1007/s10664-018-9613-x

[16] Perry, D.E., Sim, S.E., Easterbrook, S.M.: Case studies for software engineers. In: Proceedings. 26th International Conference on Software Engineering. pp. 736–738 (May 2004). https://doi.org/10.1109/ICSE.2004.1317512

[17] Poston, R.M., Sexton, M.P.: Evaluating and selecting testing tools. In: Proceedings of the Second Symposium on Assessment of Quality Software Development Tools. pp. 55–64 (1992). https://doi.org/10.1109/AQSDT.1992.205836

[18] Rafi, D.M., Moses, K.R.K., Petersen, K., Mäntylä, M.V.: Benefits and limitations of automated software testing: Systematic literature review and practitioner survey. In: 7th International Workshop on Automation of Software Test (AST). pp. 36–42 (2012). https://doi.org/10.1109/IWAST.2012.6228988

[19] Rainer, A., Hall, T., Baddoo, N.: Persuading developers to "buy into" software process improvement: a local opinion and empirical evidence. In: 2003 International Symposium on Empirical Software Engineering, 2003. ISESE 2003. Proceedings. pp. 326–335. IEEE, Rome, Italy (Sept 2003). https://doi.org/10.1109/ISESE.2003.1237993

[20] Raulamo-Jurvanen, P., Kakkonen, K., Mäntylä, M.V.: Using surveys and web-scraping to select tools for software testing consultancy. In: Abrahamsson, P., Jedlitschka, A., Nguyen, D.A., Felderer, M., Amasaki, S., Mikkonen, T. (eds.) Product-Focused Software Process Improvement. pp. 285–300. Springer International Publishing, Cham (2016). https://doi.org/10.1007/978-3-319-49094-6_18

[21] Raulamo-Jurvanen, P., Mäntylä, M.V., Garousi, V.: Choosing the right test automation tool: A grey literature review of practitioner sources. In: Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering, EASE'17. pp. 21–30. ACM, NY, USA (2017). https://doi.org/10.1145/3084226.3084252

[22] Raulamo-Jurvanen, P., Hosio, S., Mäntylä, M.V.: Practitioner evaluations on software testing tools. In: Proceedings of the Evaluation and Assessment on Software Engineering. pp. 57–66. EASE '19, ACM, New York, NY, USA (2019). https://doi.org/10.1145/3319008.3319018

[23] Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering **14**(2), 131–164 (2009). https://doi.org/10.1007/s10664-008-9102-8

[24] Seaman, C.B.: Qualitative methods in empirical studies of software engineering. IEEE Transactions on Software Engineering **25**(4), 557–572 (1999). https://doi.org/10.1109/32.799955

[25] Sjøberg, D.I.K., Dybå, T., Jørgensen, M.: The future of empirical methods in software engineering research. In: Future of Software Engineering. FOSE '07. pp. 358–378. IEEE (2007). https://doi.org/10.1109/FOSE.2007.30

[26] Stemler, S.E.: A comparison of consensus, consistency, and measurement approaches to estimating interrater reliability. Practical Assessment, Research & Evaluation **9**(4), 1–11 (2004), `https://www.ingentaconnect.com/content/doaj/15317714/2004/00000009/00000004/art00001`

[27] Taipale, O., Smolander, K., Kälviäinen, H.: Cost reduction and quality improvement in software testing. In: Software Quality Management Conference (2006)

[28] Vos, T.E.J., Marin, B., Escalona, M.J., Marchetto, A.: A methodological framework for evaluating software testing techniques and tools. In: 12th International Conference on Quality Software. pp. 230–239. IEEE (2012). https://doi.org/10.1109/QSIC.2012.16

[29] Yin, R.K.: Case Study Research: Design and Methods. SAGE Publications, Inc. (2014)