https://doi.org/10.51153/kjcis.v6i2.177

Guidelines for the Development of Automated Test Case Designing/ Generation Tool(s)

Samia Rafique[1], Rizwan Bin Faiz[2]

Abstract

Test case generation is the most intellectually demanding and labor-intensive activity in software testing. It plays the most important role in the quality of software products and reduces testing costs. Among all these automated test case generation is the most emerging and challenging area in today's research. Currently, several test case designing/generation tools exist but they are disorganized and unstructured since they are implementing various sets of parameters which is hard to categorize that's why they are still misaligned with the current requirement of the software industry. There exists a real need to have such a Test Case generation framework that aligns functions of Automated Test Case Generation tools with the current requirements of the software industry. Thus aim of this research is to provide guidelines for the development of automated test case designing/generation tools. To do so we have performed a literature survey which identified six common referred tools and existing proposed frameworks. After that, an international questionnaire-based survey had been conducted in software industries to identify their challenges related to test case generation tools and desired parameters. The results indicated that industries are not satisfied with the current tools and need more sophisticated tools. After analysis, it is concluded that software test case generation is a long process that starts from the identification of parameter sets and ends with the test procedures. It is evident from the results that all assembling test sets activities are highly supported (by all variables) by large and medium scale organizations, highly experienced quality engineers, and organizations that are certified by CMMI (at levels 4 and 5) and ISO(9001 and 90003).

Keywords: Automated Tools, Test Case Generation, parameter set, CMMI.

1 Introduction

Testing is the most integral part of the software development life cycle. It also has a great influence on the quality of the final product [10]. The testing process itself like a software development process consumes more than 50% time and cost of software development [11][12][16]. The main focus of software engineering research is on the reduction of cost and improving software quality. Automating the testing process helps to achieve these objectives and improve the overall software testing process[16]. Testers can achieve high test coverage by using automated testing tools[24].

[1]*Affiliation, Department of Computing, Riphah International University, Islamabad, Pakistan (Samiarafique7@gmail.com)*
[2]*International Islamic University, Islamabad, Pakistan (rizwan.faiz@riphah.edu.pk)*

Testing tools are the software products that are used to automate or semi-automate the software testing methods or processes, such as test case designing, execution, and incident reporting [1,25]. There are large numbers of automated testing tools available for test planning, controlling, and monitoring; all of these tools share a passive philosophy toward test case designing/ generation[12]. These tools leave crucial time and cost-consuming and demanding activities for human testers. It is not without reason that test case designing is difficult to automate with the current tools and techniques available in IT industries since the domain of possible inputs (potential test cases)even for a small program is normally too large to be exhaustively tested[12]. Since software process improvements, test automation tools, and standardization of the software testing process are the prominent issues[16].

Among all testing activities test case designing/generation is the most challenging and logically demanding task it can have a strong impact on the effectiveness of the whole software testing process[11] [12]. Different testers will understand the requirement according to their intellectual level and generate different test cases to test the same program. Over the past few decades, a large number of work on test case generation techniques and tools had been done. Automated techniques and tools for test case generation are increasingly adopted by the software industries in software testing practice but there still a big gap exists between real software application systems and the practical usability of the test case generation tools proposed by the research community[12]. A great number of tools have been developed for testing the software product but they are not mature enough. According to the survey on software test practices tool usage rate in test case designing activity is only 30% [1] [13].

Tools perform test case generation differently as they are not following any common set of guidelines. This research aims to establish a tools improvement criterion that provides useful guidelines for improving the capabilities of the tool and bridging the gap between the software industry and academia.

Testing is a systematic process, but it has limited resources and a set of testing tools. In the literature many surveys [1,14,16] on software testing tools and techniques are conducted which indicated that the automated testing area is not satisfactory; still, there is a need for sophisticated tools. There is a major gap in the software testing research community and industry practices [1,12]. Currently, several test case designing/generation tools exist [23,26-30] and they implement a variety of parameters, but they are still misaligned with the current requirement of the software industry [1]. A survey on test practices [1,14] indicated that test case designing tools need improvements and interoperability with software development. Automated test case designing/ generation tools do not develop any consensus they all have a different scope because they don't have any guideline not even a single tool know which parameters should be implemented to generate test cases this is the reason, they are not satisfying the software industry. This gap is showing that there is no systematic guidelines exist for automated test case generation tools development. There is a rigorous need to overcome this problem and bridge the gap between the software testing research community and industrial practices. To meet our research objective, we have set down a research question:

RQ: What set of parameters should be implemented in automated test case designing/generation tool(s)?

Section 2 reviews the literature while section 3 lays down details of research methods and data collection from the population i.e., the software industry. Section 4 analyses results and then based upon which proposed methodology is laid in section 5. A research paper is concluded in section 6.

2. Literature Review

Researchers in the literature have presented different frameworks for software testing tools or techniques [2][3][5]. The purpose of the literature review was to investigate how comprehensively frameworks are proposed. It also explains the detailed parameters of commonly referred automated test case generation tools. So, from the literature, we find the parameters which need to be verified by industry for test case generation frameworks. In the end gap analysis is identified and improvements are suggested for automated test case generation frameworks.

J.LEE et al [1] surveyed to identify the weakness of the software testing process and activities which are used in testing tools. Respondents were asked about the desired improvements in testing activities and processes. The survey consisted of three parts, the introduction of the company, the testing environment, and the last most important test process. The last part questions were selected based on standardized practices of the software testing process from software testing standard 29119. Major five processes of that standard were selected, test estimation, test planning, test case designing, test execution, defect management, and defect reporting. Results of the survey showed that usage of automated testing tools in the software industry is low, and there is a need for improvement in tools. Especially tools usage in test design activity is about 30% only. These survey questions cover the main activities of the standardized testing process whereas our research showed the detailed analysis of test designing activities.

Bao N.Nguyen et.al [2] presented an "Innovative framework GUITAR" for the development of automated GUI testing tools. Such a framework can be used for test case designing tools. This framework was used for developing four types of tools including the test case generation tools. Test case generation tools generate test cases based on a graph generator. It was compared with five existing frameworks on five parameters: model generation, model verification, test case generation, test oracle, and supported platform. Results showed that GUITAR support automated test case generation on multiple platforms. It has been evaluated through industrial case studies. This framework does not tell a complete process of test case generation. This is applicable only for model-based tool development. In comparison to our work, it is limited to GUI.

The framework that is proposed must have justification or selection criteria for phases of parameters for each phase and has to be validated or must have industrial feedback. In [3] established a "methodological framework for evaluating software testing tools". It was divided into five parts: The objective of the tools that define the research question for tool evaluation; the nature of the tool which contains prerequisites, performed operations, results, and tool license; a subject who was using the tool, the subject is a worker of the company; the object is the function or program under test and final part define the variables which show the efficiency, effectiveness, and satisfaction of the tools. This framework was evaluated through three case studies but it did not mention any tool name which is evaluated by this framework.

In [4] author illustrated a "framework for automated test case generation for web application testing called TASSA". The framework was designed based upon the limitations of the existing five SOA (Service Oriented Architecture) testing tools to generate test cases, organize and execute them. It was evaluated on a business process execution case study. This framework generates test cases against every assertion of the business process under test.

Tools selection criteria in the TASSA framework except TCGET were not defined. Existing tool parameters are selected based on the literature. It was only applicable to web domain applications. Y.H.Tung et al [5] worked on a framework for a web application to minimize the test cases. In this framework, test cases were designed based on the web application's data dependency and control dependency. These dependencies were extracted from the application source code. A test case generation algorithm was introduced which generates test cases based on web application dependencies. An experiment was conducted to evaluate the proposed algorithm. Results showed that generated test cases reduced the number of test cases in terms of data and control dependency.

This approach is specific to web applications. Selection criteria for the test case generation algorithm are not identified and not specified with any test case generation tool.

Cristiano et al [6] presented a test case generation framework for GUI from use case design by using the model-based testing tool. This framework writes the use case description based on test case generation goals in common natural language. The second step of the framework was to select the appropriate algorithm by using the tools and generate the test case. After that generated test cases are translated into test scripting language by using the parameterized test script. Then these test cases are imported into libraries to start the test execution.

In [7] introduced a framework for "automated black-box test case generation for java classes" which was categorized into three parts; test path generator, test data generator, and test class generator. The test path generator reads the UML diagrams and OCL (Object Constraint Language) to identify the feasible test path then the test data generator generates the test input and expected output for every test path. This framework used logical constraints which identify test paths that are used as test input and generate the expected output for the program. It was applied to a running example of Java classes to generate test cases.

The provided framework is only java specific and it is not validated by industry.

A.A.Sofokleous et al. [8] introduced a framework for automated test case generation. It was the integration of a basic program analyzer and automated test case generation (ATCG). The basic program analyzer creates a flow graph and extracts the statements for the test whereas ATGC consists of two algorithms, these algorithms search the test input space and generate test cases for every input concerning the edge coverage criteria. An experiment was conducted, and the results of the experiments showed that this framework achieves high code coverage as compared to other frameworks.

T. Abdou et al.[9] presented a framework that defines the testing process of open-source software. The open-source software development process is different from other software process. That's why the open-source software testing process is not structured, this study selects the Net Beans, Mozilla, and Apache HTTP servers as open-source software and explains their testing process. The testing process of this open-source software is compared with ISO/IEC standard software testing process. Results showed that activities and tasks of Open-source software testing are similar between the two activities and highlighted the improvements in the open-source software testing process. Machine learning and its techniques play a vital role in every expect of the domain. In [33], the author proposed to use the machine learning method mutual information technique based on the filter-based method for subband selection.

By analyzing the existing framework [3][4][5][7] and tools parameters we can see most of these studies focused on code coverage and produced test input. Some studies depended on academia and researchers select the limitations from the literature and without evaluating it from industry tried to provide the solution [32][33][34]. Besides this software industry is not satisfied with software engineering practices and tools. Some studies provide the solution, but these are not verified by industry. So it is clear there is a real need to have such a Test Case generation framework that aligns functions of Automated Test Case Generation tools with the current requirements of the software industry. Our work provides solutions that involve industrial feedback. We verify our all findings from the industry. Our work is a step towards bridging the gap between the software industry and academia. It is noticed that reviewed frameworks are domain specific whereas we focused on independent of the domain which is why our framework is generic and we have come up with such diversified phases of ATCG.

3. Research Methodology

In the literature, there are two approaches for researchers are deductive approach and inductive approach [15]. In the deductive approach some hypotheses, theories, and suggestions are created from a theoretical framework and experiments are conducted to accept or reject these hypotheses, theories, and suggestions. This approach is "Top-down" it starts from more general to more specific. While in the inductive approach researcher interprets the phenomenon which is based on the respondent's explanation. It is called the "Bottom" approach; it results to establish a new theory from data. For this purpose, systematic tools are used, like questionnaires, interviews, and theories. This analysis of the research approach provides a new theoretical framework to understand the phenomenon. So we will use an Inductive approach for our research work.

3.1. Research Method.

In the literature [15] there are five research methods case study, experiment, survey, archive analysis, and history. All these methods are different in data collection from each other and follow different procedures to collect the data. Every method is different to answer the different research questions from each other. Case study and experimental research methods are used to answer the "how" and "why" types of research questions. However, surveys are used to find the answer to "what" and "which" types of research questions. So, our research deals with the "what" and "which" types of research questions so we will use surveys for our research work to solve our problem.

3.2. Research design

Pfleeger and Kitchenharm have presented principles of designing a survey in six parts [17-22] so we have designed our survey according to these guidelines. Survey research can be done by using questionnaires or interviews. Questionnaires are the best approach for surveys so we use questionnaires to find the solution to our problem. There are many advantages of using questionnaires in a relatively short time more data can be achieved at a substantially low cost. The sample size of the research can be increased using a questionnaire. Questionnaires results can be easily analyzed. Our survey participant companies and respondents' requirements are listed below. Our survey participated companies were required to:

- Must be CMMI certified
- Must be a user of automated software testing tools

Our survey participants were required to:

- Must have at least one year of experience in software testing
- Be currently involved in software testing activities
- Can be national and international
- Every participant must understand testing tools

The object of our questionnaire was to identify the challenges, improvements, and desired parameters of automated test case generation tools. Our questionnaire was designed within three months. It consists of two parts the first part is the demographic detail of the company and respondents second part consists of parameters of automated test case generation tools at the end open-ended question was also added to take the reviews of respondents.

3.3. Data collection method and sources

Our data collection tool was questioned the scope of our study is global so data was collected from different countries. For this purpose, we have developed an online questionnaire using the "Google doc service". The population of our survey was software companies that are testing tool users.

Our primary data was collected from national and international software development companies. An instrument for gathering our required primary data was a questionnaire sent to national and international software companies. The questionnaire consists of a set of parameters of automated test case generation tools. Almost 10 countries and more than 130 respondents were engaged in our primary data collection and their responses were very valuable for our research work.

3.4. Tools evaluation

In the section of the primary study, we evaluated software test case generation/designing tools. Six commonly referred tools are selected. To ensure the quality of our work we selected tools based on their Availability and Consistency in their versions.

Availability: Tools should be accessible publically either commercial or open source.
Consistency: Tools having the latest versions from 2010 onward.

3.5. Preparation of questionnaire

We have designed our survey under some rules which are as follows:

1) We use very simple language for our questionnaire which is understandable for every respondent
2) The length of the questionnaire was not very large so those respondents don't think it's fatiguing and respond to it.
3) The closed-ended questionnaire was selected to facilitate the respondent because in open-ended questions respondents hesitate to respond it takes more time.
4) All technical words or statements are explained next to that statement or word because some terms have different names in academia and industry but with the same concept.
5) The questionnaire was designed electronically because it is more feasible for respondents to respond it does not require any form to download online URL will be shared.

3.6. Demographic detail

This section presented the demographic detail of our respondents and companies which were involved in our survey. The survey was sent to more than 200 experienced people from national and international organizations out of which we received 137 responses and 11 responses were discarded to make survey results more reliable as these responses were not properly filled up. Our research focus was not limited to Pakistan only we collected our responses from outside the border also. We have received responses from 10 different countries including the USA, UK, Australia, Egypt, Denmark, India, Tunisia, Bangladesh, UAE, and Pakistan. We have analyzed our data based on four variables that are Size of the organization, CMMI Certification, ISO Certification, and experience of respondents.

3.7. Size of Organizations

Organization size is also divided into three categories high, medium, and low based on the number of employees in each organization. Organizations that have less than 50 employees are categorized as small 1-50 "1<Emp≥50", ones that have less than 250 employees are categorized as medium "50<Emp ≥250" and organizations with more than 250 employees are categorized as large scale organizations "250<Emp". Our responded organizations included NetSole, Systems, IBM, and PCMS group.

Large ≥ 251
51 ≤ Medium ≥ 250
1 ≤ small ≥ 50

Responded organizations were certified by CMMI(Capability Maturity Model Integration) and ISO(International Organizations for Standardizations) and these organizations were at CMMI levels 3,4 and 5 and ISO certified were 9001 and 90003. The average rate of experience of respondents in software testing is 6 years and every respondent experienced a minimum of 1 year. We further divided experience into three categories i.e. high, medium, and low based on the response range. The categorization is as follows:

High>10 years
5 years< Medium≥10 years
Low≤5 years

4. Results and Findings

After analyzing our data we have discussed our results as guidelines for ATCG tools development.

Our industrial feedback ranges from 40% to 100% and we categorized our data into High, Medium, and Low.

100 < High> 80
80 < Medium>60
60 <Low> 40

There are a few parameters that support 79% so as an exceptional case we consider them in the high category.

Table 4.1: Parameters set identification for generating test cases

How are parameter sets identified in your organizations?	**Organization size**		**CMMI**		**ISO Certification**		**Experience**
	Large	Medium	Level 5	Level 4	9001	90003	High
Identify parameter sets	High	High	High	High	High	High	High
Combine parameter sets	High	High	High	Medium	High	High	High
Prioritize parameter sets	High	High	High	High	High	High	High
Record parameter sets	High	High	High	High	Medium	High	High
Establish traceability b/w parameter sets	High	High	High	High	High	High	High

Table 4.1 shows that activities of feature set identification are highly supported by large and medium-scale organizations and highly experienced quality engineers. Whereas organizations that are certified by CMMI (at levels 4 and 5) and ISO(9001 and 90003) also highly support only two features are supported medium. It is therefore integrated into the proposed guidelines.

Table 4.2: Test conditions identification for test case Design

How are test conditions identified in your organizations?	**Organization size**		**CMMI**		**ISO Certification**		**Experience**
	Large	**Medium**	**Level 5**	**Level 4**	**9001**	**90003**	**High**
Identify Test conditions	High	High	High	High	High	High	High
Prioritize test conditions	High	High	High	High	High	High	High
Record test conditions	High	High	Medium	Medium	High	High	High
Establish traceability b/w test conditions	High	High	High	High	High	High	High

Table 4.2 shows that all parameters of test conditions are highly supported by large and medium-scale organizations, ISO-certified organizations, and highly experienced quality engineers.

Organizations that are certified by CMMI (at levels 4 and 5) also support all parameters highly except Record test conditions which gain medium support. It is therefore integrated into the proposed guidelines.

Table 4.3: Coverage item identification for Test Case Design

How are test	**Organization size**		**CMMI**		**ISO Certification**		**Experience**
coverage items	Large	Medium	Level 5	Level 4	9001	90003	High
Derive Test organizations?							
derived in your Coverage items	High	High	High	High	High	High	High
Prioritize test coverage items	High	High	High	High	High	High	High
Record test coverage items	High	High	High	High	High	High	High
Establish traceability b/w test coverage items	High	High	High	High	High	High	High

Table 4.3 illustrate that all features of test coverage items are highly supported (by all variables) by large and medium-scale organizations, highly experienced quality engineers, and organizations that are certified by CMMI (at levels 4 and 5) and ISO(9001 and 90003). It is therefore integrated into the proposed guidelines.

Table 4.4: Designing/ generating test cases

How are test cases	**Organization size**		**CMMI**		**ISO Certification**		**Experience**
derived in your	Large	Medium	Level 5	Level 4	9001	90003	High
organizations?							
Dynamic symbolic execution	Medium	Medium	High	Medium	High	High	Low
Random-based	Low	Low	Low	Low	Medium	Low	High
Search -based	Low	Low	Medium	Low	Low	Low	Low
Prioritize test cases	High	High	High	High	High	High	High
Record test cases	High	High	High	High	High	High	High
Establish traceability in test cases	High	High	High	High	High	High	High

As table 4.4 explains that in test case generation/designing last three parameters are highly supported by all variables. Whereas dynamic symbolic execution, Random-based, and search-based test case generation gain medium and low support by all variables.

Table 4.5: Assembling test cases into test sets

How are test sets assembled in your organizations?	Organization size		CMMI		ISO Certification		Experience
	Large	Medium	Level5	Level4	9001	90003	High
Assemble test sets	High	High	High	High	High	High	High
Prioritize test sets	High	High	High	High	High	High	High
Record test sets	High	High	High	High	High	High	High

Table 4.5 illustrated that all assembling test sets activities are highly supported (by all variables) by large and medium scale organizations, highly experienced quality engineers, and organizations that are certified by CMMI (at levels 4 and 5) and ISO(9001 and 90003). It is therefore integrated into the proposed guidelines.

Table 4.6: Deriving test procedures

How are test procedures derived in your organizations?	Organization size		CMMI		ISO Certification		Experience
	Large	Medium	Level5	Level4	9001	90003	High
Derive Test procedures	High	High	High	High	High	High	High
Prioritize test procedures	High	High	High	High	High	High	High
Record test procedures	High	High	High	Medium	High	High	High
Establish traceability b/w test procedures	High	High	High	High	High	High	High

The above table 4.6 illustrate that all parameters of test procedures are highly supported by large and medium-scale organizations, highly experienced quality engineers, and organizations which are certified by CMMI (at levels 4 and 5) and ISO (9001 and 90003) except record procedures are gain medium support at CMMI level4 organizations. It is therefore integrated into the proposed guidelines.

5. Proposed Framework

The ultimate objective of this research is to Provide Guidelines for the development of automated test case designing/generation tools”. We also have identified the current start of the art in software test case generation tools and frameworks.

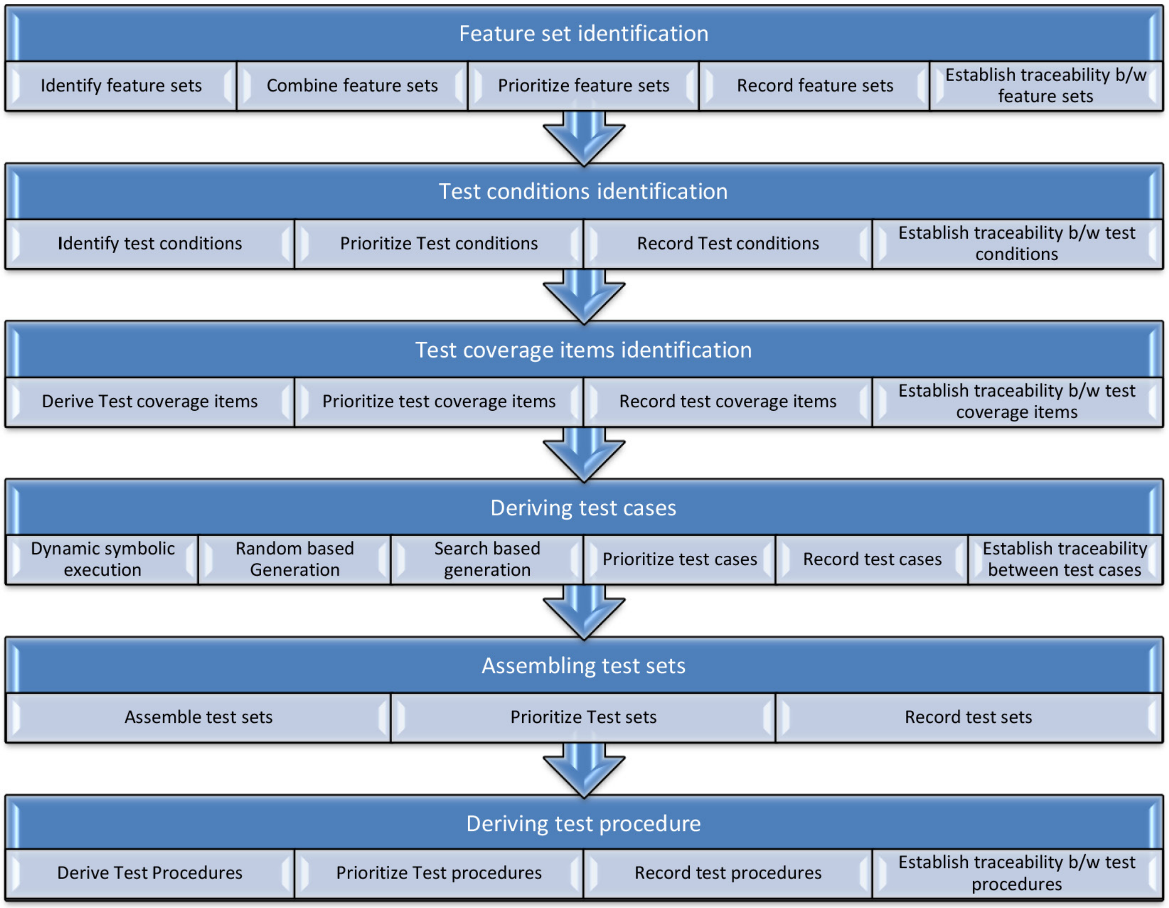
Feature set identification
Identify feature sets
Combine feature sets
Prioritize feature sets
Record feature sets
Establish traceability b/w feature sets
Test conditions identification
Identify test conditions
Prioritize Test conditions
Record Test conditions
Establish traceability b/w test conditions
Test coverage items identification
Derive Test coverage items
Prioritize test coverage items
Record test coverage items
Establish traceability b/w test coverage items
Deriving test cases
Dynamic symbolic execution
Random based Generation
Search based generation
Prioritize test cases
Record test cases
Establish traceability between test cases
Assembling test sets
Assemble test sets
Prioritize Test sets
Record test sets
Deriving test procedure
Derive Test Procedures
Prioritize Test procedures
Record test procedures
Establish traceability b/w test procedures

Figure 4.1 Proposed Framework

We have conducted an online international questionnaire-based survey to identify the real need for software quality department people. It took three months to prepare and launch the survey. In addition, we have tried to discover the long-term vision of automated software test case generation tools. So that software developers can manage and improve their software testing process (tools). After the data analysis, it is concluded that test case generation is a long process starting from parameter set identification, test conditions identification, deriving test coverage items designing test cases and assembling test cases into test sets, and then ending on test procedure gains consistent support from the large organizations, highly experienced quality engineers and CMMI and ISO certified organizations. Our survey questions were designed in a detailed way, and no one mentioned any new activity for test case generation except mentioned. For future work, it is recommended that a tool should develop by following these guidelines, and then an experiment can be conducted with this tool and also on already proposed tools in large-scale organizations on different projects. Then analyze and compare both tools and evaluate their results based on quality metrics. We then proposed a framework that will help the software industries in the context of software testing tool(s) improvements. So, our empirical study gives guidelines to software developers in the identification of parameter sets required by developers of testing tools. Proposed a framework that is designed specifically for international industry feedback by organizations that are at CMMI levels 3,4,5 ISO 9000 and 90003, respondents having high experience, and organizations that have large numbers of employees. Net sole, systems, and Microsoft. Such a study will be helpful in the development of test case execution tool(s) as well since they are meant to solve problems of the industry if they are specifically designed based upon industrial requirements.

6. Conclusion

Software test case generation and testing tools are the most researched area now a day. Many organizations which are going to adopt test case generation tools are still facing the challenges such as too many test cases regarding one input, more human involvement, and almost half of the development cost consumed in this phase. Since existing frameworks were not validated by the industry, and some were not clearly defined on which bases they are proposed. Therefore, current research decomposed parameters of test case generation into hierarchical structure beginning with a high-level quality goal followed by its corresponding sub characteristics which provide comprehensive guidelines which improve the test case generation tools development. Six tools have been selected based upon their availability and maturity and identified their process of test case generation and then we highlighted their limitations. We first identified important activities and their corresponding tasks of test case generation and then parameter set for the development of test case design tools. Such a study will be helpful in the development of test case execution tool(s) as well since they are meant to solve problems of the industry if they are specifically designed based upon industrial requirements.

Reference

[1] Lee, J., S. Kang, and D. Lee. "Survey on software testing practices." *IET Software 6,* no. 3 (2012): 275-282.

[2] Nguyen, Bao N., Bryan Robbins, Ishan Banerjee, and AtifMemon. "GUITAR: an innovative tool for automated testing of GUI-driven software." *Automated Software Engineering* 21, no. 1 (2014): 65-105.

[3] Vos, Tanja EJ, Beatriz Marin, Maria Jose Escalona, and Alessandro Marchetto. "A methodological framework for evaluating software testing techniques and tools." *In Quality Software (QSIC), 2012 12th International Conference on,* pp. 230-239. IEEE, 2012.

[4] Stoyanova, Vera, DessislavaPetrova-Antonova, and Sylvia Ilieva. "Automation of Test Case Generation and Execution for Testing Web Service Orchestrations." In *Service-Oriented System Engineering (SOSE), 2013 IEEE 7th International Symposium on,* pp. 274-279. IEEE, 2013.

[5] Tung, Yuan-Hsin, Shian-Shyong Tseng, Tsung-Ju Lee, and Jui-FengWeng. "A novel approach to automatic test case generation for web applications." In *Quality Software (QSIC), 2010 10th International Conference on,* pp. 399-404. IEEE, 2010.

[6] Bertolini, Cristiano, and AlexandreMota. "A framework for GUI testing based on use case design." In *Software Testing, Verification, and Validation Workshops (ICSTW), 2010 Third International Conference on,* pp. 252-259. IEEE, 2010.

[7] Hu, Yi-Tin, and Nai-Wei Lin. "Automatic black-box method-level test case generation based on constraint logic programming." In *Computer Symposium (ICS), 2010 International,* pp. 977-982. IEEE, 2010.

[8] Sofokleous, Anastasis A., and Andreas S. Andreou. "Automatic, evolutionary test data generation for dynamic software testing." *Journal of Systems and Software* 81, no. 11 (2008): 1883-1898.

[9] Abdou, Tamer, Peter Grogono, and PankajKamthan. "A Conceptual Framework for Open Source Software Test Process." In *Computer Software and Applications Conference Workshops (COMPSACW), 2012 IEEE 36th Annual,* pp. 458-463. IEEE, 2012.

[10] Kasurinen, Jussi, OssiTaipale, and Kari Smolander. "Analysis of problems in testing practices." In *Software Engineering Conference, 2009. APSEC'09. Asia-Pacific,* pp. 309-315. IEEE, 2009.

[11] Anand, Saswat, Edmund K. Burke, TsongYueh Chen, John Clark, Myra B. Cohen, Wolfgang Grieskamp, Mark Harman, Mary Jean Harrold, and Phil McMinn. "An orchestrated survey of methodologies for automated software test case generation." *Journal of Systems and Software* 86, no. 8 (2013): 1978-2001.

[12] Vos, Tanja EJ, Arthur I. Baars, Felix F. Lindlar, Peter M. Kruse, Andreas Windisch, and Joachim Wegener. "Industrial scaled automated structural testing with the evolutionary testing tool." In *Software Testing, Verification and Validation (ICST), 2010 Third International Conference on,* pp. 175-184. IEEE, 2010.

[13] Ng, S.P., Murnane, T., Reed, K., Grant, D., Chen, T.Y.: 'A preliminary survey on software testing practices in Australia'. Proc. Conf. Australian Software Engineering, 2004, pp. 116–125

[14] Garousi, Vahid, and JunjiZhi. "A survey of software testing practices in Canada." *Journal of Systems and Software* 86, no. 5 (2013): 1354-1376.

[15] Wohlin, Claes, Per Runeson, Martin Höst, Magnus C. Ohlsson, BjörnRegnell, and Anders Wesslén. *Experimentation in software engineering. Springer Science & Business Media,* 2012.

[16] Kasurinen, Jussi, OssiTaipale, and Kari Smolander. "Software test automation in practice: empirical observations." *Advances in Software Engineering* 2010 (2010).

[17] Pfleeger, Shari Lawrence, and Barbara A. Kitchenham. "Principles of survey research part 2: designing a survey." *Software Engineering Notes* 27, no. 1 (2002): 18-20.

[18] Kitchenham, Barbara A., and Shari Lawrence Pfleeger. "Principles of survey research: part 3: constructing a survey instrument." *ACM SIGSOFT Software Engineering Notes* 27, no. 2 (2002): 20-24.

[19] Pfleeger, Shari Lawrence, and Barbara A. Kitchenham. "Principles of survey research: part 1: turning lemons into lemonade." *ACM SIGSOFT Software Engineering Notes* 26, no. 6 (2001): 16-18.

[20] Kitchenham, Barbara, and Shari Lawrence Pfleeger. "Principles of survey research part 4: questionnaire evaluation." *ACM SIGSOFT Software Engineering Notes* 27, no. 3 (2002): 20-23.

[21] Kitchenham, Barbara, and Shari Lawrence Pfleeger. "Principles of survey research: part 5: populations and samples." *ACM SIGSOFT Software Engineering Notes* 27, no. 5 (2002): 17-20.

[22] Kitchenham, Barbara, and Shari Lawrence Pfleeger. "Principles of survey research part 6: data analysis." *ACM SIGSOFT Software Engineering Notes* 28, no. 2 (2003): 24-27.

[23] Galler, Stefan J., and Bernhard K. Aichernig. "Survey on test data generation tools." *International Journal on Software Tools for Technology Transfer* 16, no. 6 (2014): 727-751.

[24] Rafi, Dudekula Mohammad, Katam Reddy Kiran Moses, Kai Petersen, and Mika V. Mäntylä. "Benefits and limitations of automated software testing: Systematic literature review and practitioner survey." In *Proceedings of the 7th International Workshop on Automation of Software Test,* pp. 36-42. IEEE Press, 2012.

[25] Abran, Alain, James W. Moore, Pierre Bourque, Robert Dupuis, and L. Tripp. "Guide to the software engineering body of knowledge, 2004 version." *IEEE Computer Society* 1 (2004).

[26] Csallner, Christoph, and YannisSmaragdakis. "JCrasher: an automatic robustness tester for Java." *Software: Practice and Experience* 34, no. 11 (2004): 1025-1050.

[27] Wahid, M., and A. Almalaise. "JUnit framework: An interactive approach for basic unit testing learning in Software Engineering." In *Engineering Education (ICEED),* 2011 3rd International Congress on, pp. 159-164. IEEE, 2011.

[28] Fraser, Gordon, and Andrea Arcuri. "Whole test suite generation." *Software Engineering, IEEE Transactions on* 39, no. 2 (2013): 276-291.

[29] Fraser, Gordon, and Andrea Arcuri. "EvoSuite at the SBST 2013 Tool Competition." In *ICST Workshops,* pp. 406-409. 2013.

[30] De Castro, Andreza MFV, Gisele A. Macedo, Eliane F. Collins, and Arilo C. Dias-Neto. "Extension of Selenium RC tool to perform automated testing with databases in web applications." In *Proceedings of the 8th International Workshop on Automation of Software Test,* pp. 125-131. IEEE Press, 2013.

[31] Alim, Affan, et al. "The most discriminant subbands for face recognition: A novel information-theoretic framework." *International Journal of Wavelets, Multiresolution, and Information Processing* 16.05 (2018): 1850040.

[32] Bork, D., Roelens, B. A technique for evaluating and improving the semantic transparency of modeling language notations. *Softw Syst Model* 20, 939–963 (2021).

[33] Cognitive Impairment and Dementia Data Model: Quality Evaluation and Improvements Dessislava Petrova-Antonova * and Sophia Lazarova GATE Institute, Faculty of Mathematics and Informatics, Sofia University “St. Kliment Ohridski”, 1504 Sofia, Bulgari Received: 15 November 2022 / Revised: 6 January 2023 / Accepted: 27 January 2023 / Published: 30 January 2023.

[34] Brady SS, Brubaker L, Fok CS, et al. Development of Conceptual Models to Guide Public Health Research, Practice, and Policy: Synthesizing Traditional and Contemporary Paradigms. *Health Promotion Practice.* 2020;21(4):510-524.