Test case quality as perceived in Sweden

Anders Adlemo
Jönköping University - School of Engineering
Jönköping, Sweden
anders.adlemo@ju.se

He Tan
Jönköping University - School of Engineering
Jönköping, Sweden
he.tan@ju.se

Vladimir Tarasov
Jönköping University - School of Engineering
Jönköping, Sweden
vladimir.tarasov@ju.se

ABSTRACT

In order to reach an acceptable level of confidence in the quality of a software product, testing of the software is paramount. To obtain "good" quality software it is essential to rely on "good" test cases. To define the criteria for what make up for a "good" test case is not a trivial task. Over the past 15 years, a short list of publications have presented criteria for "good" test cases but without ranking them based on their importance. This paper presents a non-exhaustive and non-authoritative tentative list of 15 criteria and a ranking of their relative importance. A number of the criteria come from previous publications but also from discussions with our industrial partners. The ranking is based on results collected via a questionnaire that was sent out to a limited number of randomly chosen respondents in the Swedish software industry. This means that the results are more indicative than conclusive.

KEYWORDS

Good software test cases, quality criteria, requirement-driven development, test-driven development

ACM Reference Format:
Anders Adlemo, He Tan, and Vladimir Tarasov. 2018. Test case quality as perceived in Sweden. In *RET'18: RET'18:IEEE/ACM 5th International Workshop on Requirements Engineering and Testing, June 2, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3195538.3195541

1 INTRODUCTION

The preliminary study presented in this paper forms part of a bigger research project named OSTAG (Ontology-based Software Test cAse Generation). The aim of the project has been to develop models for the (semi-)automatic production of software test cases based on requirements written in natural language. At the core in the creation of the test cases is found a semantic model, known as an ontology. The ontology is a formal model of the requirements and through the application of what is known as "inference rules" on the ontology, test cases have been produced. Some of the results from the OSTAG project can be found in [7], [11], [12], [13], [16]. One challenge when automatically producing test cases is the possibility to determine the quality of the test cases. This has been the driving force behind the preliminary study presented in this paper.

As long as software code is written by humans, bugs will undoubtedly find their way into the code. The way to handle these bugs is to undertake some kind of testing of the code. An important part in testing is made up of test cases whose main attribute is to detect the aforementioned bugs [15]. The way to procure "good" software is thus to procure "good" test cases. The problem is, what defines a "good" software test case. That is why a description of what constitutes a good test case would be important to the software community in general, and to test developers and test executioners in specific. The contribution of this paper is the presentation of a list of so called "good" software test criteria that are the result of previous definitions by other researchers, combined with input from testing experts coming from the industrial partners in the OSTAG project. The paper includes some preliminary results of how a limited number of randomly chosen Swedish experts in the software testing domain rank the different criteria.

2 RELATED WORK

The definition of what constitutes a "good" test case is not something easily captured. This observation could be one of the reasons why not too many publications can be found on the topic. The first intent to define what good test cases really entail was undertaken by Kaner in 2003 [8]. Already in the abstract he stressed that the difficulty to write good software test cases depend on the complexity to construct them. The complexity, according to Kaner, comes from three sources:

- Different types of tests are more effective for different classes of information.
- No test case will be good for all good test case criteria.
- Good domain tests are different from good, risk-based tests.

But before even trying to define criteria for good test cases, it is necessary to outline what a test case really is. The ISO-IEC-IEEE 24765-2010 standard [2] states that a test case is

- a set of test inputs, execution conditions, and expected results developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement. IEEE 1012-2004 Standard for Software Verification and Validation. 3.1.31-32 [1].
- documentation specifying inputs, predicted results, and a set of execution conditions for a test item. IEEE 1012-2004 Standard for Software Verification and Validation. 3.1.31-32 [1].

Another defacto standard is Test Maturity Model Integration (TMMi) release 1.0 [14], that states that a test case is

RET'18, June 2, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5749-4/18/06…$15.00
https://doi.org/10.1145/3195538.3195541

- a set of input values, execution preconditions, expected results and execution post conditions, developed for a particular objective or test condition, such as to exercise a particular program path or to verify compliance with a specific requirement.

As can be observed, both standards are very similar in their wording. Accordingly to Kaner, a test case is a question that you ask of the program. The point of running a test is to obtain information, for example whether the program will pass or fail the test. But the main objective, and something that everyone in software industry seem to agree upon, is for a test case to detect as many software bugs as possible in the tested code. Kaner continues his paper by presenting some criteria for what he believes a good test case should do:

(1) Find defects
(2) Maximize bug count
(3) Block premature product releases
(4) Help managers make ship / no-ship decisions
(5) Minimize technical support costs
(6) Assess conformance to specification
(7) Conform to regulations
(8) Minimize safety-related lawsuit risk
(9) Find safe scenarios for use of the product
(10) Assess quality
(11) Verify correctness of the product
(12) Assure quality

As can be observed from the list, the descriptions or criteria, on an individual level, are very different in scope, ranging from concrete activities, such as to *maximize the bug count*, to more elusive goals, such as to *minimize the safety-related lawsuit risk*. Such a diversity in the scope of the criteria is something that aggravates the work of a testing expert. What is difficult to explain is the individual importance of the different criteria, which is most important and should be handled first, and who within an organization should perform the verification of the outcome.

Atmadja et al. propose a more limited list of only four criteria [4]. These, however, are criteria that all could be verified by a tester and not by high management. Accordingly to the article, a good test case should:

(13) Have a clear purpose
(14) Focus on testing only a few aspects of the test subject[1]
(15) Produce output which can be easily verified by the tester
(16) Give reproducible errors

An important aspect that can be observed in the the list by Kaner is that some of the criteria are close to impossible to measure. For example, the criteria *find safe scenarios for use of the product*, is difficult to measure or to even assign a metric. Having stated that, an intent to measure the quality of test cases is described in [9]. The main contribution of the paper was the discovery that not only the test suites need to be taken into account for testing benchmarks but also the environment of the test: development process, problem domain, test technique and other artifacts.

[1]Test subject could range from a piece of code (in unit testing) to a general functionality (in system testing).

A recent publication by Bowes et al. presents a list of 15 testing principles that capture the essence of testing goals and best practices from a quality perspective [5]. In their conclusions, the authors specifically stress the importance of tests being correct, maintainable and simple, three criteria that were also identified as important by the authors of this paper.

Two different software development approaches are *requirement-driven development* and *test-driven development*. In relation to testing, these are sometimes known as test-last versus test-first [6]. In the first case, the requirements specifications are finished and the project is code complete before testing begins. In the second case, test cases are defined and testing executed before the completion of the object code. Which of the two approaches that provide the best test cases depend on who you ask and in what domain that person works. An experiment aimed at measuring and comparing the quality of test cases produced in a requirement-driven development environment against the quality of test cases produced in a test-driven development environment can be found in [6]. No conclusive results demonstrating any statistical difference in quality was presented, though. Another study from 2014 described an intent to extract the best features of each of the two different development approaches, thus creating a hybrid testing process [10]. The preliminary study presented in continuation have a focus on requirement-driven development and standard testing practices, as defined in ISO/IEC/IEEE 29119 Software testing standards [3]. This is important to stress because what defines a "good" test case criterion differs between the requirement-driven development domain and the test-driven development domain.

3 THE STUDY

Based on the work by Kaner [8], Atmadja et al. [4], and input from the three partner companies in the OSTAG project, 15 criteria, that from the authors perspective are candidates of "good" test cases, were formulated and then evaluated and ranked by a limited group of randomly chosen Swedish software experts. Some of the "reused" criteria from Kaner or Atmadja et al. have been renamed. In some cases, the definition of a criterion is a subset of the definition of a criterion from Kaner or Atmadja et al. and in other cases it is a superset. The numbers within parentheses in the following list correspond to the numbering of the criteria by Kaner and Atmadja et al. in Sect. 2.

- Powerful: a good test case is more likely to discover bugs than a not so good test case (1,2)
- Clear: a new tester should not require any help to execute a test case (5)
- Efficient: a test case should not require too many resources regarding staff or hardware/software (5)
- Complete: a set of test cases for a specific SW product should ensure 100% requirement coverage (6)
- Traceable: a test case could be traced to its corresponding requirement (6)
- Correct: a test case should not include any ambiguity in its execution (13)
- Simple: a test case should not include unnecessary steps or words (14)
- Compact: a test case should cover only 1-2 requirements (14)

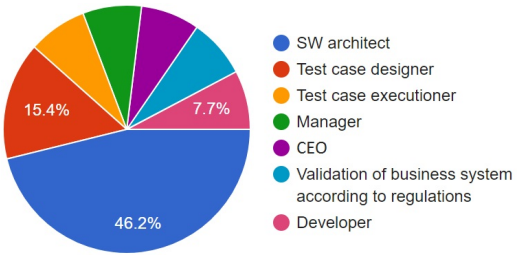

Figure 1: Relation to SW development and testing

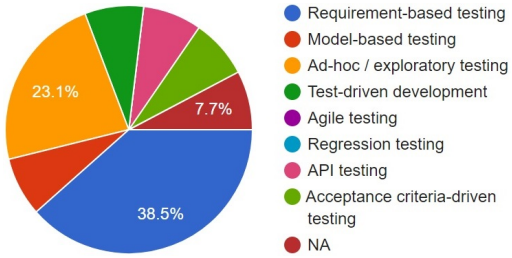

Figure 2: Type of testing performed

- Accurate: the outcome of a test case should be predictable and acquirable (15)
- Repeatable: a test case should produce the same result if run multiple times (16)
- Consistent: similar test cases should be written in a consistent fashion (0)
- Covering: a set of test cases for a specific SW product should ensure 100% code coverage (0)
- Independent: a test case should be able to execute without depending on other test cases for its success (0)
- Maintainable: a test case should be relatively easy and inexpensive to handle (0)
- Reusable: a test case could be reused, if necessary (0)

A (0) indicates a "new" criteria. As can be observed in the list, there are five "new" criteria, not covered by Kaner or Atmadja et al., i.e. *Consistent, Covering, Independent, Maintainable*, and *Reusable*. These criteria were decided to be added after discussions with our three industrial partners. Numbers (3), (4), (7), (8), (9), (10), (11) and (12) are not in the list as they, accordingly to the authors of this paper, belong to a completely different organizational level altogether and should not be expected to be handled and/or solved during testing. For example, *Help managers make ship/no-ship decisions*, should be handled by management, not by testers. Another example is *Assure quality*; quality is something that has to be build into the development process and cannot be left until the end, during the testing phase. Finally, the criterion *Powerful* is a composition of two other criteria, *Find defects* and *Maximize bug count*, as they both relate to the same thing, bugs. As can be observed from the list, the style of the criteria follow that of Atmadja et al., i.e. they are focusing on activities and metrics that can be handled by a test designer and not by a top manager, as was the case for some of the criteria defined by Kaner.

The next step in the preliminary study was to have the criteria evaluated and ranked by Swedish software experts using a Google-forms questionnaire. The range that was used to measure the conformity among the experts with regard to the presented criteria, went from 0 to 10, where 0 indicates "not at all relevant" and 10 indicates "extremely relevant". The link to the questionnaire[2] was sent out to approximately 150 individuals working in companies dealing with software development and software testing. The companies were chosen from several Swedish web-lists providing information about mostly small and medium-size companies in the software domain. So far, only 13 individuals answered the invitation, thus only providing an indication of the importance of the different criteria and not statistically proven results. This should be taken into consideration when analyzing the results in table 1. The results indicate that 53.8% of the participants in the preliminary study had a previous experience of 5 to 10 years while 30.8% had more than 10 years of experience. The distribution of the experts background is shown in figure 1. The experts relation to software development and testing is presented in figure 2.

As a part of the questionnaire, the experts were invited to comment on the criteria. Some of their remarks are shown in continuation:

- Repeatable: *With the same input it should be repeatable, but a test case outcome might vary depending on if the input is exactly the same or slightly different. If used in regression testing, you do want the same result from the same input time after time after time.*
- Accurate: *It's rare that the the outcome is predictable, it varies on many factors as quality of requirements, maturity of SW, satellite systems and integrations.*
- Correct: *If the goal is to test a strict set of instructions you would like it to be very clear. Sometimes you want it to be a bit vague so the executioner might take different ways to a specific goal and make decisions of themselves.*

Table 1: Ranking of good test case criteria

Rank	Criteria	Mean
1	Repeatable	9.2
2	Accurate	8.4
3	Correct	8.1
4	Powerful	7.5
5	Maintainable	7.4
6	Complete	7.2
7	Traceable	7.2
8	Consistent	6.8
9	Reusable	6.5
10	Simple	6.3
11	Efficient	6.3
12	Clear	6.2
13	Independent	5.8
14	Covering	5.6
15	Compact	4.5

[2]https://goo.gl/forms/MqmRNBohwEXxng022

- Powerful: *It really depends on the goal of the test case, is it to proof that happy path works, or is to find severe bugs. It's rather rare that you can combine those two.*
- Maintainable: *Maintenance is costly and test cases should be as easy as possible to maintain. Ideally, you can change a few parameters when necessary and not going through too many lines.*
- Complete: *The goal is, of course, to reach 100% requirement coverage. However, sometimes you need to focus on the areas of largest risk.*
- Traceable: *In the cases where you have clear requirements it is of great value if it is traceable to the requirements. In those cases where you have vague or non requirements or similar documentation you can't reach traceability to the requirements, but in those cases it is important that you have documented the test so it is traceable regarding to what steps you have made.*
- Consistent: *If the test cases are consistent they are much easier to combine and the cost of switching between test cases (context switch) becomes much lower.*
- Reusable: *Time is a rare resource which makes reusable test cases very attractive.*
- Simple: *What is simple varies a lot so deciding what is a unnecessary step can be very hard. Sometimes you need several small steps to make it easy to understand and to catch the right result, sometimes fewer and simplified steps is what is most feasible.*
- Efficient: *Since time is a rare resource and availability of HW and SW for execution should be as easy to accommodate as possible, efficiency is of essence.*
- Clear: *A clear test case is very valuable since the cost to introduce the executioner to the test case can be kept fairly low.*
- Independent: *With independent test cases you are more likely to be able to execute them in a feasible way. They are also easier to use as building blocks to build flows.*
- Covering: *Depending on how well requirements and code are documented, it might be very hard to get 100% code coverage. Often, this is made by the developers by using specific tools, and even then it is rare to get more than 80% code coverage.*
- Compact: *If time and resources allow you, it is good to have compact test cases that are possible to use as building blocks.*

4 CONCLUSIONS

Most of the criteria not covered by Kaner or Atmadja et al. (marked with a (0) in the previous list), received a fairly high ranking by the experts, as shown in table1. The comments by the experts also support this statement. For example, *Maintainable*, received a mean value of 7.4 and one expert pointed out that *only having to change a few parameters when necessary and not having to go through too many lines is something to strive for.* Another example, *Clear*, received a mean value of 6.2 and one comment was that *a clear test case would lower the threshold between the test designer and the test executioner.*

Three of the criteria that is normally associated with requirement-driven software development and testing, i.e. *Complete, Covering* and *Traceable*, end up behind other types of criteria. This could be attributed to the fact that many software developers, both of stand-alone software products and embedded systems, want and need more answers from their tests than just a simple "test passed". The "top" criteria, *Repeatable*, receives 9.2 on the ranking scale. Possibly a surprise to some but maybe not in the light of the trend, moving towards test-driven development where a need for repeatability is of utmost importance.

Regarding the more "managerial" criteria that were not included in this preliminary study, a diligent reader might argue that as managers and CEOs answered the questionnaire, as indicated in figure 1, they would be biased and assign lower grades to the not so "managerial" criteria. This did not occur, though, as their individual grading did not indicate any tendency towards that.

To improve the quality of the results in this preliminary study, a more extensive study needs to be undertaken in the future.

ACKNOWLEDGMENTS

The work presented in this paper was funded by the Knowledge Foundation in Sweden, grant KKS-20140170. The authors also want to thank the anonymous peer reviewers for their comments.

REFERENCES

[1] IEEE Standards Association. 2004. Software verification and validation - IEEE 1012-2004. (2004), 1–80.
[2] IEEE Standards Association. 2010. Systems and software engineering - Vocabulary ISO/IEC/IEEE 24765-2010. (2010), 1–418.
[3] IEEE Standards Association. 2013. Software and systems engineering - Software testing - ISO/IEC/IEEE 29119-2013. (2013).
[4] Michael Atmadja and Richard Zhang Shuai. 2011. Chapter 3: Testing. In *A Fresh Graduate's Guide to Software Development Tools and Technologies*, Damith C. Rajapakse (Ed.). School of Computing, National University of Singapore, 1–21. http://www.comp.nus.edu.sg/~seer/book/2e/
[5] David Bowes, Tracy Hall, Jean Petrić, Thomas Shippey, and Burak Turhan. 2017. How good are my tests?. In *Emerging Trends in Software Metrics (WETSoM), 2017 IEEE/ACM 8th Workshop on*. IEEE, 9–14.
[6] Adnan Čaušević, Daniel Sundmark, and Sasikumar Punnekkat. 2012. Test case quality in test driven development: a study design and a pilot experiment. In *Proceedings of the EASE 2012*. IET, 223–227. https://doi.org/10.1049/ic.2012.0029
[7] Muhammad Ismail. 2016. Ontology learning from software requirements specification (SRS). In *European Knowledge Acquisition Workshop*. Springer, 251–255. https://doi.org/10.1007/978-3-319-58694-6_39
[8] Cem Kaner. 2003. What is a good test case?. In *Software Testing Analysis & Review Conference (STAR) East, Orlando, FL*. 1–16. http://kaner.com/pdfs/GoodTest.pdf.
[9] Christian Pfaller, Stefan Wagner, Jörg Gericke, and Matthias Wiemann. 2008. Multi-dimensional measures for test case quality. In *Software Testing Verification and Validation Workshop, 2008. ICSTW'08. IEEE International Conference on*. IEEE, 364–368.
[10] Syed Muhammad Ali Shah, Cigdem Gencel, Usman Sattar Alvi, and Kai Petersen. 2014. Towards a hybrid testing process unifying exploratory testing and scripted testing. *Journal of Software: Evolution and Process* 26, 2 (2014), 220–250.
[11] He Tan, Anders Adlemo, Vladimir Tarasov, and Mats E. Johansson. 2017. Evaluation of an application ontology. In *Proceedings of the Joint Ontology Workshops 2017 Episode 3: The Tyrolean Autumn of Ontology, JOWO 2017*. CEUR-WS.org, 1–14. http://ceur-ws.org/Vol-2050/DEW_paper_1.pdf
[12] He Tan, Muhammad Ismail, Vladimir Tarasov, Anders Adlemo, and Mats Johansson. 2016. Development and evaluation of a software requirements ontology. In *7th International Workshop on Software Knowledge-SKY 2016 in conjunction with the 8th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management-IC3K 2016*. SCITEPRESS, 11–18.
[13] Vladimir Tarasov, He Tan, Muhammad Ismail, Anders Adlemo, and Mats Johansson. 2016. Application of inference rules to a software requirements ontology to generate software test cases. In *OWL: Experiences and Directions–Reasoner Evaluation*. Vol. 10161. Springer, 82–94.
[14] Erik van Veenendaal and Brian Wells. 2012. *Test Maturity Model Integration TMMi*. Uitgeverij Tutein Nolthenius, The Netherlands.
[15] Tsuneo Yamaura. 1998. How to design practical test cases. *IEEE software* 15, 6 (1998), 30–36.
[16] Ole Zimmermann. 2017. *Modelling complex software requirements with ontologies*. Master's thesis. Universität Rostock. Part of the OSTAG project.