Publication IV

Juha Itkonen, Mika V. Mäntylä, and Casper Lassenius. 2009. How do testers do it? An exploratory study on manual testing practices. In: Proceedings of the Third International Symposium on Empirical Software Engineering and Measurement (ESEM 2009). Lake Buena Vista, Florida, USA. 15-16 October 2009. IEEE. Pages 494-497. ISBN 978-1-4244-4842-5.

How Do Testers Do It? An Exploratory Study on Manual Testing Practices

Juha Itkonen, Mika V. Mäntylä and Casper Lassenius
Helsinki University of Technology, Software Business and Engineering Institute
P.O. Box 9210, FI-02015 TKK, Finland
juha.itkonen@tkk.fi, mika.mantyla@tkk.fi, casper.lassenius@tkk.fi

Abstract

We present the results of a qualitative observation study on the manual testing practices in four software development companies. Manual testing practices are seldom studied, and based on the literature we conjecture that they have a strong effect on the effectiveness of manual testing. We observed testing sessions of 11 software professionals performing system level functional testing. As a result we identified 22 manual testing practices that we classified into 9 test session strategies and 13 detailed test execution techniques. Many of the identified techniques were based on similar ideas as traditional test case design techniques. However, the subjects applied these techniques during manual testing without separate test design phase. The results indicate that software professionals use a wide set of strategies and techniques when performing manual testing. Testers seem to need and use techniques even if applying exploratory testing.

1. Introduction

Revealing defects efficiently is one of the great challenges in software engineering. In practice, detection of functional defects at the system level is still largely dependent on the contribution of human testers doing manual testing. Most new defects are found by manual testing, and test automation is often seen as a way of removing the enactment of simple and repetitive tasks from human testers in order to free up time for creative manual testing, not to replace it [1,2].

We consider test execution a key activity in the manual testing process, and think that more research is needed to understand the state of the practice of all manual testing activities in real software development organizations. Testing is traditionally considered a process that relies upon executing test cases that are carefully designed using test case design techniques [3,4]. However, test execution is typically not a simple mechanic task consisting of executing completely specified test cases. Instead, testers' skill and knowledge are important also during test execution. Previous research shows that individual aspects such as testers' skills have as strong an effect on the results of testing as do the test case design techniques [5]. Studies on industrial practice and practitioner reports show that rigorous and thoroughly documented test-case based testing is not common in industry [1,6,7] and many researchers and practitioners, see e.g. [8-12], have emphasized the role of experience and skills in software testing activities.

One experience-based alternative to test-case based testing is exploratory testing (ET) approach that does not rely on the documentation of test cases prior to test execution. ET builds on applying the experience and knowledge of the tester during manual testing by performing simultaneous test design, execution, reporting, and learning of the tested application [6,8]. Practitioner reports of exploratory testing approaches claim them to be effective and cost-efficient [8-10]. In addition, some scientific studies, e.g. [11,13], as well as our own research [6,12] support the hypothesis that experience-based testing could be effective and cost-efficient in suitable contexts.

There is an identified need for research on the actual practice on software testing [14,15]. However, the majority of prior works focuses on test case design and research on manual testing as a whole is rare. In this paper, we report the first results of an observation study of software testing work in industry with the purpose of shedding some light upon what testers really do when performing experience-based manual testing.

2. Research goals and methods

The main goal of this study is to increase understanding of how testers perform experience-based and exploratory manual testing in the context of system level functional testing. This study aims at identifying

978-1-4244-4841-8/09/$25.00 ©2009 IEEE

and classifying the actual practices that testers use. With 'practices' we mean any manual testing behaviour that follows some logic, has some goal, and involves utilising testers experience or skills, i.e., behaviour that is not only following a written script.

This study is exploratory in nature, as opposed to theory development. In order to properly study these human aspects of software engineering, we chose to use qualitative inquiry using field observations as the primary data collection method [16]. Field observation was selected as a method to gain access to the actual testing tasks that the subjects performed in their natural working environments. By using direct observations, instead of interviews or inquiries afterwards, we avoided having to rely on descriptions and conceptualizations by the subjects, based upon their recollection of how they work. To gain better understanding of what the observed tester was doing, how, and why, we applied the "think-aloud" approach [16], meaning that we asked the subjects to think aloud, i.e., describe what they were doing and thinking during the testing session. However, the subjects typically verbalized their work quite briefly and discontinuously.

This study was carried out in four case organizations. All organizations were Finnish small to medium sized companies in the software product business. The case companies were selected based on accessibility, i.e., the case companies were accessible to researcher through research co-operation. The total number of tested software systems in the observed sessions was six. Two of the six products were developed with a separate testing organization, and also the observed subjects in these cases were from the testing organization. The rest of the products were developed in organizations without an independent testing team. In these cases the actual testing tasks were carried out by people in varying roles in the organization.

The total number of observed sessions was 11, i.e., one observation session for each subject. The lengths of the observed sessions varied between 1.5 and 2.5 hours. All the observed sessions were part of the subjects' normal testing activities and took place in the same environment the subject would normally do the testing work. The focus of the observations was narrowed to the tester's practices and behaviour during the testing session.

The observed subjects were selected using purposeful sampling [16] among software development professionals who had functional testing as one of their duties. We selected subjects with different roles and backgrounds. We aimed at finding information-rich cases ([16] p. 242), in order to get a wide view of the variety of practices professionals use in testing work. Four of the eleven observed professionals were primarily testers. Rest of the subjects were not full time testers, but rather application area experts in such roles as product and project management, customer support, and software development. The subjects were experienced in their application domain with an average of 11 years of domain experience and 7.8 years of software engineering experience.

The observations were carried out by the first author alone. The data was collected by taking thorough field notes during the test session observation using a pre-planned structure. In addition, if test documentation, such as high level test cases or other guidance, was used during the observed sessions it was recorded to support data analysis.

The observations were augmented by short (15-30 min) interviews after each observation session. The interview data was used as demographic background information and to understand if the observed session was somehow special or differs from the normal testing activities in the organization.

The field note data was analyzed by coding and categorizing the findings. We first used a pre-defined initial coding scheme where we coded the data using high level codes that were based on our research questions and general theory of software testing. The preliminary list of codes was refined and extended during analysis work. After coding we formed categories of the practices based on the abstraction level the practice worked at and similarity of the practices.

3. Results

In this chapter, we describe the results of this study. We describe the classification that was created in order to better understand the large amount of identified testing practices. Few selected practices are described to give examples of different types of identified practices. In addition, we outline the challenges testers encountered in the observation sessions.

We identified that practices worked at different abstraction levels and analysed the findings from this viewpoint. Based on the analysis we divided the practices into two main classes, namely *test session strategies* and *test execution techniques*. The two main classes were further divided into subclasses based on the similarity as described in Table 1. The classifications are described in the next subsections.

We also identified challenges related to experience-based testing approach. Despite of the numerous test execution practices that we identified, the subjects lacked practices for logging and tracking of testing; transferring the requirements knowledge to testers and utilizing it; and focusing testers' attention to ensure the most important aspects of the tested features are tested. In experience-based testing applicable methods are needed for documentation, knowledge transfer, and

978-1-4244-4841-8/09/$25.00 ©2009 IEEE

tracking, when the test-case based documentation practices are not applied.

3.1. Test Session Strategies

The test session strategies are high level practices that the subjects used to give an overall structure to their testing work. Session strategies gave the general guideline of how to proceed in testing and what aspects to cover. In combination with session strategies, subjects typically used free exploratory testing or some test execution technique (see Table 1) to test the details of each feature. The session strategies are further divided into subclasses of *exploratory strategies* and *documentation based strategies*.

The exploratory strategies were used to guide and give structure to exploratory testing. An example of such strategy is *User interface exploring* where a tester structured the testing through user interface features and proceeds from feature to feature and covers all UI features, but relied on experience-based approach in testing each individual feature. Another example is *Exploring weak areas* that relied on the tester's experience and tacit knowledge of potential weak areas, not based on documented analysis. Rest of the strategies are *Aspect oriented testing, Top-down functional exploring, Simulating a real usage scenario,* and *Smoke testing by intuition and experience*.

The documentation based session strategies, in contrast to exploratory strategies, are used to guide experience-based testing using documented tests or other documentation. The used documentation can be test-cases that are described on varying level, or release notes and defect reports. The documentation is used as a checklist to give structure for test execution or as high-level test-cases that are extended and deepened by experience-based approach. An example of such strategy is *Data as test cases* where test data was documented and used to give structure for testing and manage coverage. Instead of functional steps the testing was guided by situations defined in terms of test data. The strategy was used in testing a financial system where the test data represented different kinds of customers with different properties, services, and personal situations. This data set was used for covering a representative set of situations in experience-based testing of the features of the system. Other strategies are *Exploring high-level test cases*, and *Checking new and changed features*.

3.2. Test Execution Techniques

The test execution techniques are practices that the subjects used for testing individual or tightly related features, or such small details as input values. We categorized the techniques into three subclasses: *Exploratory, comparison* and *input techniques*.

Table 1. Classification of the identified practices

Test session strategies	Exploratory	6 practices
	Documentation based	3 practices
Test execution techniques	Exploratory	6 practices
	Comparison	4 practices
	Input	3 practices

The exploratory techniques are techniques that are used for exploring one isolated functionality or a single function. Most of the exploratory techniques are based on hypothesis of a certain type of defects that the technique aims to reveal, or a certain typical situation where defects are often revealed. For example, *Simulating abnormal and extreme situations* is a technique for testing extreme situations or scenarios. The aim is to evaluate how a function performs on and beyond its limits and reveal problems of handling abnormal and stress situations where the limits or rules of normal operation are violated. Other techniques are *Testing alternative ways, Exploring against old functionality, Persistence testing, Feature interaction testing*, and *Defect based exploring*.

The comparison techniques are used for evaluating the test outcomes and making difference between correct, expected; and incorrect, erroneous behaviour during testing, i.e., how to recognize a defect. Subjects needed comparison techniques to analyse complicated features and evaluate the expected outcome especially when the correct function was not unambiguously specified or involved many aspects that must be taken into consideration. For example, *Comparing within the software* is a technique for comparing similar features in different places of the same system. The aim is to assess if a feature works correctly or not by investigating the consistency of functionality inside a software. Other techniques are *Comparing with another application or version, Checking all the effects*, and *End-to-end data check*.

The input techniques are used for detailed testing of individual input values or set of related inputs. Input techniques are used to manage covering the details of a feature and selecting relevant test cases or values for testing. For example, *Testing boundaries and restrictions* is a technique that focuses on testing the boundary values and other restrictions of input data. The aim is to cover all explicit and implicit restrictions of a single function and reveal defects that are associated with handling boundaries and restrictions. Other techniques are *Testing input alternatives* and *Covering input combinations*.

978-1-4244-4841-8/09/$25.00 ©2009 IEEE

4. Discussion and conclusions

It is important to notice that many of the identified practices are actually based on theories and assumptions that are same or similar to some of the traditional test-case design techniques. As an example the techniques in the *Input technique* category were similar to the classic equivalence class partitioning and boundary value analysis techniques [3]. In addition, the *Covering input combinations* was a simple technique that captured the basic idea of the combinatorial testing. Many of the exploratory strategies and techniques were similar to the general heuristics, rules of thumb, and experience-based lessons found in software testing textbooks [3,10]. However, the difference between our findings and how the techniques are presented in the literature is that the execution-time practices were used as part of test execution, not as test design methods beforehand. The identified comparison techniques seem to be techniques that are not often described in testing literature, because in test-case based testing the comparison to documented expected results does not require separate techniques. As a notable exception, Kaner et al. list some evaluation-based techniques including comparison techniques and consistency heuristics [10].

As a conclusion we state that this study provides initial results of a research on the manual testing practices in the form testing is practiced in industry. We identified 22 manual testing practices used by professionals. We created a classification of the testing practices to help better understand the numerous findings and support future research. This study supports the hypothesis that testers, in practice, apply numerous techniques and strategies during test execution and do not mechanically rely on test documentation. Testers need testing techniques even if applying experience-based and exploratory testing approaches. Finally, we identified that execution-time techniques are partly similar to test-case design techniques, but are strongly experience-based and applied in the non-systematic fashion during test execution.

We continue research on manual testing practices in real testing organizations and include more dedicated testing organizations in our research and focus to understanding what makes testers good. It is important to study how the testing practices can be trained and how the practices can benefit experience-based testing without sacrificing other important aspects of testing, such as planning, tracking and coverage.

5. References

[1] Andersson, C. and P. Runeson, "Verification and validation in industry - a qualitative survey on the state of practice," Proceedings of International Symposium on Empirical Software Engineering, 2002, pp. 37-47.

[2] Berner, S., R. Weber, and R.K. Keller, "Observations and Lessons Learned from Automated Testing," Proceedings of International Conference on Software Engineering, 2005, pp. 571-579.

[3] Myers, G.J., The Art of Software Testing, New York: John Wiley & Sons, 1979.

[4] Beizer, B., Software Testing Techniques, New York: Van Nostrand Reinhold, 1990.

[5] Juristo, N., A.M. Moreno, and S. Vegas, "Reviewing 25 years of Testing Technique Experiments," Empirical Software Engineering, vol. 9(1-2), 2004, pp. 7-44.

[6] Itkonen, J. and K. Rautiainen, "Exploratory testing: a multiple case study," Proceedings of International Symposium on Empirical Software Engineering, 2005, pp. 84-93.

[7] Ahonen, J.J., T. Junttila, and M. Sakkinen, "Impacts of the Organizational Model on Testing: Three Industrial Cases," Empirical Software Engineering, vol. 9(4), Dec. 2004, pp. 275-296.

[8] Bach, J., "Exploratory Testing," The Testing Practitioner, E. van Veenendaal, ed., Den Bosch: UTN Publishers, 2004, pp. 253-265.

[9] Våga, J. and S. Amland, "Managing High-Speed Web Testing," Software Quality and Software Testing in Internet Times, D. Meyerhoff, B. Laibarra, R. van der Pouw Kraan, and A. Wallet, eds., Berlin: Springer-Verlag, 2002, pp. 23-30.

[10] Kaner, C., J. Bach, and B. Pettichord, Lessons Learned in Software Testing, New York: John Wiley & Sons, Inc., 2002.

[11] Beer, A. and R. Ramler, "The Role of Experience in Software Testing Practice," Proceedings of Euromicro Conference on Software Engineering and Advanced Applications, 2008, pp. 258-265.

[12] Itkonen, J., M.V. Mäntylä, and C. Lassenius, "Defect Detection Efficiency: Test Case Based vs. Exploratory Testing," Proceedings of International Symposium on Empirical Software Engineering and Measurement, 2007, pp. 61-70.

[13] Houdek, F., T. Schwinn, and D. Ernst, "Defect Detection for Executable Specifications — An Experiment," International Journal of Software Engineering & Knowledge Engineering, vol. 12(6), Dec. 2002, p. 637.

[14] Martin, D., J. Rooksby, M. Rouncefield, and I. Sommerville, "'Good' Organisational Reasons for 'Bad' Software Testing: An Ethnographic Study of Testing in a Small Software Company," Proceedings of International Conference on Software Engineering, 2007, pp. 602-611.

[15] Juristo, N., A. Moreno, S. Vegas, and F. Shull, "A Look at 25 Years of Data," Software, IEEE, vol. 26(1), 2009, pp. 15-17.

[16] Patton, M.Q., Qualitative Research and Evaluation Methods, Thousand Oaks: Sage, 2002.

978-1-4244-4841-8/09/$25.00 ©2009 IEEE