DE GRUYTER OPEN
RIGA TECHNICAL UNIVERSITY
Information Technology and Management Science
ISSN 2255-9094 (online)
ISSN 2255-9086 (print)
December 2017, vol. 20, pp. 48–53
doi: 10.1515/itms-2017-0008
https://www.degruyter.com/view/j/itms

Test Case Review Processes in Software Testing

Oksana Petunova[1], Solvita Bērziša[2]
[1, 2] *Riga Technical University, Latvia*

***Abstract* – Qualitative system requirements and thoughtful communication are the key factors to successful implementation of software development projects. However, errors that occur by misunderstanding and incomprehension between parties involved in the project can lead not only to the high cost of the project but also to a large number of defects discovered only after the system release, which in turn strongly influences the product quality. The software review processes are implemented to reduce projects costs and ensure high product quality. The goal of the present paper is to identify the role of review processes in software testing. To achieve this goal, the process of test case review has been implemented during testing.**

***Keywords* – Communication, inspection, software review process, software testing, static testing.**

I. Introduction

Successful use of any software depends on users' satisfaction and its usefulness. In order to ensure high quality and maximum efficiency of the daily users' work, any software should operate continuously and uninterrupted.

Software testing plays an important role in the software development lifecycle [2]. Software testing is a process rather than a single activity. This process starts with test planning, designing test cases, preparing for execution, evaluating a status and ends with the test closure. During software testing, the developed software is tested for compliance with the international standards, requirements and business needs. If discrepancy and shortcomings are found in software during the testing, it means there are defects and errors which should be fixed.

According to the seven principles of testing, it is irrational and impossible to check all combinations of software inputs and preconditions and find all defects in the software product [7]. The software testing can provide an ability to reduce the number of undetected defects in the developed software. Even if defects are not found, testing cannot prove that software is 100 % defect free [6]. In order to organise effective and useful software testing, it is necessary to choose appropriate testing techniques. Test cases should also be defined based on project and product risks [1], [6].

Many errors and defects are discovered at the end of the testing or are not discovered at all until users find them after the release of the software [2], [9], [12]. Low software quality can reduce product reputation and increase the possibility that end users and customers prefer using competitors' services [1], [2], [5], [9].

Defects which are discovered at the end of testing and defects which are found by software users are much more expensive to fix than defects which are discovered at the earliest project phases when, for example, business requirements were defined. This immediately increases project costs. Fixing such defects is more expensive and labour-intensive because it is necessary to fix defects not only in a software code, but also in the documentation where this functionality is described [2], [5]. In order to decrease the number of defects detected at the end of testing or after the release of the software, it is necessary to take preventive measures starting with the project earliest phases. One of these preventive measures is static testing [6], [10].

Static testing provides a good opportunity to improve software quality and reliability. Its techniques provide a possibility to get early feedback on software quality. The static testing techniques can be used without a computer because software testing is performed without software executing. That is why static testing is convenient to use at the project earliest phases. One of the static testing techniques is software review process [6].

The goal of the present paper is to identify the role of software review processes in software testing. To achieve this goal, one of the software review types – a test case review – has been introduced in software testing for the case study project.

The case study has been conducted on the basis of the following tasks: to identify the role of communication and test case reviews in the software testing; to find out if timely information availability and the test case reviews can improve software quality; to determine if the test case reviews can reduce total testing time.

The paper is organised as follows: Section 2 describes theoretical foundation and literature review of related studies. Section 3 describes case study process, a project used for the case study and data collected for the analysis. Section 4 summarises the results of the test cases review implementation. Section 5 describes the implications of these results and the conclusions that may be drawn.

II. Related Research

In general, software review process is one of the few methods that can be used for error detection and correction in the software development process. The achievements of this process can be associated with a possibility to detect and fix defects at the projects earliest phases when defect prevention costs are low in comparison with the projects latest phases [6], [8], [9], [11].

Software review process is one of the most effective and productive methods how to assess and verify the quality of the software and its artefacts in the software development lifecycle. It is one of static testing (verification) techniques, and its main objective is to find and avoid an error which appears during the software development process. Software review process can be applied to any artefact created during the software development

lifecycle, for example, business requirements, system designs, code, test plans, test cases and documentation [10], [12]. A systematic evaluation of these documents and artefacts occurs during this process. One of main software review process benefits is that it can be used at the earliest software development project phases [1], [5], [6]. There are many types of software review processes. The most popular types are formal and informal reviews, inspection, walkthrough, technical reviews, peer reviews and management reviews. Despite the fact that all types of the software review have the same objective, there are differences between their implementation processes [6].

Software formal reviews follow a formal process, which is implemented in accordance with a structured and regulated procedure. The most common type of formal reviews is software inspection. Software inspection is the most comprehensive software review process that is implemented in strict compliance with the procedure, and it is led by the trained moderators. Software formal reviews, especially software inspection, consist of six main steps: planning, kick-off, preparation, review meeting, rework and follow-up [1], [11]. During these six steps, the following activities take place: review session planning and preparation for it, analysis of the software artefacts and documents, review meeting, result analysis and evaluation. More detailed information about software formal reviews and software inspection can be found in the article "State-of-the-Art: Software Inspections after 25 Years" [1] and in the *ISTQB* [6] book.

Software informal reviews, in turn, are not documented and can be applied many times during the software development process without any structured preparation and organisation [6]. The most common type of informal reviews is a walkthrough. Walkthroughs are led by the software artefacts or document authors. An author guides the participants through the documents according to his/her thought process to achieve a common understanding and gather feedback in order to improve software product and artefacts [6].

Technical reviews vary from quite informal to very formal reviews and they are led by a trained moderator or a technical expert. It is often performed as a peer review without management participation. Technical reviews have the same objectives as walkthroughs. In both cases, it is the fastest and less expensive way how to get feedback about software, software artefacts and its quality in general [6].

The software reviews are based on communication between project development team members and all project stakeholders [1], [6]. Communication and software reviews are the most important processes which should be taken into account not only at the software testing phase, but also during the all software development life cycle. Communication is a "two-way" process of information exchange, in which information is received and understood by both participants [6]. Thoughtful communication is a gold key to successful project implementation. Poor communication can lead to confusion and misunderstanding.

In the software engineering literature, there are articles which describe software review process as one of the most important and effective processes in the software development lifecycle. For example, *Frank Elberzhager*, *Jürgen Münch* and *Danilo Assmann* in [2] conducted a study to identify relationships between software reviews and software testing. In the course of their work, three approaches how to define and direct the testing process using defect detection at the project earliest phases were analysed. As a result of this study, it was proved that there were relationships between software testing and software reviews. Information obtained in software review can be used to define and direct the software testing process.

In turn, *Anurag Goswami*, *Gursimran S. Walia* and *Urvashi Rathod* in [3] described how it was possible to improve an ability of individual participants involved in the software review process to detect defects because the success of the software reviews strictly depended on qualification and skills of participants involved. As a result of this study, it was proved that was necessary to keep in mind qualifications and skills of participants in the software reviews to obtain maximum results.

Aybuke Autum, *Hakan Petersson* and *Claes Wohlim* in [1] summarised theoretical information about software review process and software inspections within 25 years of its introduction. They described development areas of software reviews and inspections, new methods that were invented during this time, and advantages and disadvantages of the software inspections. Chris Kemerer and Mark Paukl in [5] investigated how the software review in the coding phase could impact and improve quality of the software.

III. Case Study Design and Collected Data

Many studies describe the benefits of the software reviews which occur during the requirements analysis and development [3]–[5]. The present paper describes benefits of the test case reviews which occur in software testing. The following research questions are raised:

- What role is played by communication and test case reviews in software testing?
- Can timely information availability and test case reviews improve software quality?
- Can test case reviews reduce the total testing time?

The case study has been conducted at the company which deals with the development, support, testing and maintenance of telecommunication and information technology services in nine European countries: Latvia, Estonia, Lithuania, Sweden, the Netherlands, Germany, Austria and Croatia. The study has been conducted within one department which is responsible for support and development of billing system "XXX". New billing system releases have been discussed and examined in the study. A new release of the billing system for one country rolls out every 4–8 months. Since there are nine countries in total, there is roll-out of one or two new system releases each month. Here it is useful to note that business requirement analysis, software testing, installation and software maintenance are performed by department employees, but software design, development and coding are implemented by an outsourcing company. The case study analyses only software testing, including test planning, analysis of business requirements and solution description, test case design and testing of new features as well as system

preparation for a new release. Figure 1 displays the structure of software testing phase. For each release, it is planned to carry out a 4-week testing phase.

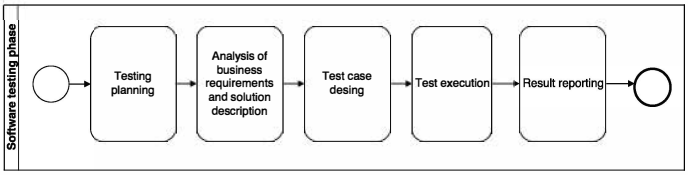

Fig. 1. Software testing phase.

The case study has been organised into 3 steps:

1. Collecting statistics of 4 release testing before introducing the test case review;
2. Implementing the process of test case review;
3. Collecting statistics of 4 release testing when test case reviews have been used.

The statistics were collected based on communication between testers and developers, business analysts, support specialists, and weekly reports that have been delivered to stakeholders within a year. Detailed summary of case study steps and collected data are provided further in this section.

Step 1. Collection of Statistics (before)

Table I and Fig. 2 summarise the quantity and complexity of defects found during testing of four releases before the test case reviews have been implemented. It is important to note that Table I contains information about the defects which have been found in the first new system release testing. It means that if another country is rolling out a similar system release version, then only regressive testing is carried out.

All defects are divided into three levels of complexity: critical defects, major defects and minor defects. A critical level is assigned to a defect, which completely hampers or blocks the system functionality. A major level is assigned to a defect, which occurs when the functionality is functioning grossly away from the expectations or not doing what it should be doing, but a minor level is assigned to a graphic or grammatical defect.

The data analysis of defects shows that a total of 206 defects have been found, where 25 % of them are critical defects, 62 % – major defects and 13 % – minor defects. The average number of defects found in the release is 51 defects: 13 critical defects, 32 major defects and 6 minor defects.

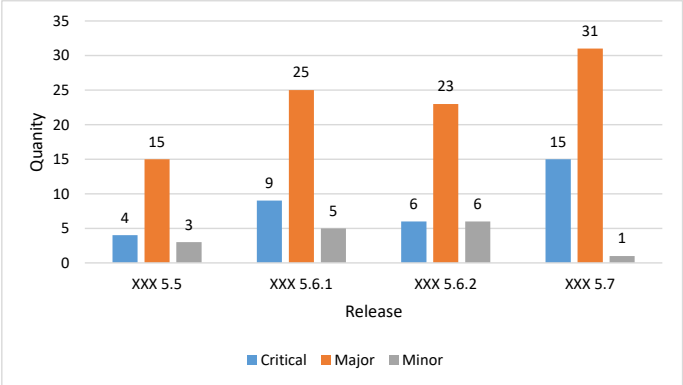

Fig. 2. Defect distribution by complexity, before software review process implementation.

Step 2. Implementation of the Test Case Review Process

The test cases review process has been implemented in software testing for the case study project. During the test cases reviews have been checked if test cases are developed in accordance with defined requirements and standards. Several checkpoints have been defined to support the review process. Example of checkpoints is given in Table II. The test case reviews are performed by another tester who did not write them, but who has the same qualification as an author, or even better, e.g., a test lead or a business analyst.

Table IV summarises defects, errors, gaps and problems and their occurrence reasons that have been found in the test cases reviews. Many defects were timely discovered and eliminated without any extra costs using information discovered in the test case reviews. The information also enables testers to enhance their knowledge and better develop future test cases, which also affect quality and speed of testing in the future software releases. A total of 17 test case review sessions have been organised during an evaluation period.

TABLE I

NUMBER OF DEFECTS BEFORE SOFTWARE REVIEW PROCESS IMPLEMENTATION

Release	**Number of defects and its degree of complexity**				**Total**
	1 week	*2 week*	*3 week*	*4 week*	
XXX 4.8	12 (critical) 20 (major) 3 (minor)	8 (critical) 15 (major)	3 (major) 5 (minor)	2 (minor)	68
XXX 5.2	10 (critical) 18 (major) 1 (minor)	18 (major)	1 (critical) 4 (major) 4 (minor)	3 (major) 2 (minor)	62
XXX 5.3	3 (critical) 12 (major)	7 (major) 2 (minor)	2 (major) 4 (minor)	0	30
XXX 5.4	11 (critical) 10 (major)	6 (critical) 10 (major)	5 (major) 2 (minor)	1 (critical) 1 (major)	46

TABLE II
CHECKLIST FOR TEST CASE REVIEW

No.	Checklist	Assessment (Yes/No)	Remarks
1.	Is the correct test case template being used?		
2.	Is the following information correct? • References to business requirements; • information about the author; • creation time; • an idea on the test cases; • preconditions for test cases execution.		
3.	Was a product risk factor taken into account when test case execution conditions were defined?		
4.	Are the test cases able to cover all defined requirements?		
5.	Are external areas, which could affect the implementation of the requirement, identified and included in the test cases?		
6.	Are equivalence classes identified? Are all possible equivalence classes included in the test cases?		
7.	Are test data identified and included in the test cases?		
8.	Are boundary values, negative and invalid values identified and included in the test cases?		
9.	Are negative scenarios included in the test cases?		
10.	Are test steps defined in a correct and logical sequence?		
11.	Are the expected results defined for all test steps?		
12.	Is the expected result correctly identified?		
13.	Are test cases free of grammatical errors?		
14.	Are test cases developed consistently with use cases?		

Step 3. Collection of Statistics (After)

Table III and Fig. 3 summarise the number of defects and complexity of release testing after the test cases review process has been implemented.

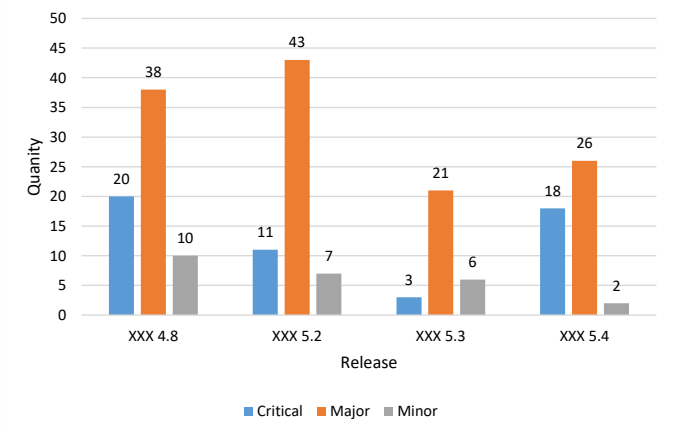

Fig. 3. Defect distribution by complexity, after software review process implementation.

The data analysis of defects shows that 145 defects have been found, 23 % of them are critical defects, 65 % – major defects and 11 % – minor defects. The average number of defects found in the release is 36 defects: 8 critical defects, 24 major defects and 4 minor defects.

IV. RESULTS

Statistical data analyses before and after implementation of the test case review process show that the total number of defects has been reduced by 30 %. The approximate percentage of critical, major and minor defects did not change but decreased a possibility of finding defects at the end of the testing when test execution was almost completed. Most of defects are found during the first 2 weeks. Further, in this section the results of the formulated research questions are discussed.

A. What Role Is Played by Communication and Test Case Reviews in Software Testing?

The reviews are based on communication among not only project development team members, but also among all project stakeholders. In testing, the test case reviews can help detect errors in the test cases and also business requirements.

Timely elimination of defects reduces total project costs because finding and fixing defects at the earliest project phases are much cheaper than fixing such defects at the latest project phases [2]. During the test case review, many problems with business requirements were found. As the errors in the business requirements are defect occurrence reasons in later software products, it is necessary to think of a possibility of organising business requirements reviews. If the developer has the opportunity to check the test cases while implementing a code,

it is possible that this will help implement codes that may cause potential defects. During the test case reviews, problems with communication between testers and developers were found. As developers were external service providers and communication was held in English and mostly by emails, sometimes it took more time to explain to developers where defects and problems were. Table IV shows defects and mistakes found during the test case reviews and their occurrence. Many defects are fixed without any additional costs.

B. Can Timely Information Availability and Test Case Reviews Improve Software Quality?

Information is the main exchange material in the project and team interaction. The ongoing exchange of information allows quickly finding necessary solutions and taking important decisions. Lack of information or incorrect information can cause misunderstanding among project team members. Misunderstanding of information increases the possibility of taking an incorrect decision or leads to errors in many project documents. Therefore, the lack of communication and information is one of the defect occurrence reasons in the software development.

The availability of information can reduce confusion among project team members and reduce the possibility of taking incorrect decisions. Software reviews are based on communication and availability of information. Timely detection of errors and failures and their elimination during software reviews increase software quality by reducing the possibility of finding defects after software release.

During test case reviews, problems with information availability were found. Often important information is not timely passed to the tester, so many errors and defects in the test cases occur in that regard. Testers sometimes use incorrect and outdated requirements and solution description versions, which in turn can result in some mistakes.

C. Can Software Reviews Reduce the Total Testing Time?

Partially, software reviews can reduce the testing time which is required to verify if defects are fixed because many errors and failures are already eliminated during the test case reviews. The total testing time is influenced not only by the test cases reviews, but also by the human factor, force majeure, emergency situations, the delivery time of new functionality etc.

TABLE III
NUMBER OF DEFECTS, AFTER SOFTWARE REVIEW PROCESS IMPLEMENTATION

Release	**Number of defects and its degree of complexity**				**Total**
	1 week	*2 week*	*3 week*	*4 week*	
XXX 5.5	4 (critical) 6 (major)	9 (major) 2 (minor)	1 (minor)	0	22
XXX 5.6.1	9 (critical) 11 (major)	10 (major)	4 (major) 2 (minor)	3 (minor)	39
XXX 5.6.2	5 (critical) 12 (major)	1 (critical) 11 (major)	5 (minor)	1 (minor)	35
XXX 5.7	10 (critical) 16 (major)	5 (critical) 13 (major)	1 (minor) 2 (major)	0	47

TABLE IV
MOST OFTEN DETECTED DEFECTS AND THEIR OCCURRENCE REASONS

Defect	**Occurrence reason**
Incomplete test cases	Insufficient knowledge of the developed functionality or software
Lack of negative test cases	Insufficient knowledge of the software testing (methods, techniques, principles etc.)
	Lack of data in the requirements
	Changes to the requirements after test case development was finished
Lack of test data	Lack of data in the requirements
Inappropriate and incorrect test data	Insufficient knowledge of the developed functionality or software
Incorrect expected results	Insufficient knowledge of the developed functionality or software
Grammatical mistakes	No check of the existence of grammatical errors was made
Incomplete test execution results	Data about the expected results and defects are not updated after each test execution
Information about defects is not updated	Data about the expected results and defects are not updated after each test execution
Test cases are not updated if changes are made to requirements	Lack of communication. Information about changes in requirements is not passed to testers

V. Conclusion

In this case study, the test case review process has been organised and implemented in software testing. Implementation of software reviews is one of the project success factors. During the test cases reviews, many errors are detected and corrected not only in the test cases, but also in the requirements and software code, which in turn affect software quality and project costs. As a result, the number of critical defects and a total number of defects for separate system releases are reduced [2], [5], [12].

The review process of project documents (requirements, system designs, software code, test plans, management plans etc.) helps improve and achieve maximum results. Defects found during test case reviews helped localise problematic areas in the software development lifecycle [3]–[5].

To obtain a maximal result from reviews, it is necessary to take into account participants' skills and knowledge, especially qualifications, skills and knowledge when selecting reviewers. The reviewers who are inexperienced, i.e., do not know business logic and system functionality, cannot find errors and can even make new mistakes. Similar conclusions are made in [3], [9], [11].

In general, the results and the accuracy of analysis obtained in the paper are limited to the defined case study. To interpret the obtained knowledge as a general approach that can be used in software testing, additional case studies on other projects are necessary. Regarding this case study, in future the reviews are planned to be implemented in the business requirement analysis and development processes.

References

[1] A. Aurum, H. Petersson, and C. Wohlin, "State-of-the-art: software inspections after 25 years," *Software Testing, Verification and Reliability*, vol. 12, no. 3, pp. 133–154, 2002. https://doi.org/10.1002/stvr.243

[2] F. Elberzhager, J. Münch, and D. Assmann, "Analyzing the relationships between inspections and testing to provide a software testing focus," *Information and Software Technology*, vol. 56, no. 7, pp. 793–806, Jul. 2014. https://doi.org/10.1016/j.infsof.2014.02.007

[3] A. Goswami, G. S. Walia, and U. Rathod, "Using Learning Styles to Staff and Improve Software Inspection Team Performance," *2016 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, Oct. 2016. https://doi.org/10.1109/issrew.2016.38

[4] K. Holl and F. Elberzhager, "Mobile Application Quality Assurance: Reading Scenarios as Inspection and Testing Support," *2016 42th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, Aug. 2016. https://doi.org/10.1109/seaa.2016.11

[5] C. F. Kemerer and M. C. Paulk, "The Impact of Design and Code Reviews on Software Quality: An Empirical Study Based on PSP Data," *IEEE Transactions on Software Engineering*, vol. 35, no. 4, pp. 534–550, Jul. 2009. https://doi.org/10.1109/tse.2009.27

[6] A. Spillner, T. Linz and H. Schaefer, *Software Testing Foundations: A Study Guide for the Certified Tester Exam*, Rocky Nook, 2007.

[7] "What are the principles of testing?" [Online]. Available: http://istqbexamcertification.com/what-are-the-principles-of-testing/

[8] Q. Shan, G. Rong, H. Zhang, G. Liu, and D. Shao, "An Empirical Evaluation of Capture-Recapture Estimators in Software Inspection," *2015 24th Australasian Software Engineering Conference*, Sep. 2015. https://doi.org/10.1109/aswec.2015.17

[9] H. Potter, M. Schots, L. Duboc, and V. Werneck, "InspectorX: A game for software inspection training and learning," *2014 IEEE 27th Conference on Software Engineering Education and Training (CSEE&T)*, Apr. 2014. https://doi.org/10.1109/cseet.2014.6816782

[10] F. Salger, G. Engels, and A. Hofmann, "Inspection effectiveness for different quality attributes of software requirement specifications: An industrial case study," *2009 ICSE Workshop on Software Quality*, May 2009. https://doi.org/10.1109/wosq.2009.5071552

[11] N. Hashemitaba and S. H. Ow, "Generative Inspection: An Intelligent Model to Detect and Remove Software Defects," *2012 Third International Conference on Intelligent Systems Modelling and Simulation*, Feb. 2012. https://doi.org/10.1109/isms.2012.48

[12] D. L. Parnas and M. Lawford, "The role of inspection in software quality assurance," *IEEE Transactions on Software Engineering*, vol. 29, no. 8, pp. 674–676, Aug. 2003. https://doi.org/10.1109/tse.2003.1223642

Oksana Petunova has earned her *BSc*. degree in Information Technology from Riga Technical University, Latvia (2015). The current position is Information System Test Specialist at Tele2 SSC. She holds ISTQB Certified Tester certificate.
E-mail: milisenta@inbox.lv

Solvita Bērziša holds a Doctoral Degree and is an Assistant Professor at the Institute of Information Technology of Riga Technical University (Latvia). She obtained her *Dr. sc. ing.* (2012), *BSc.* (2005) and *MSc.* (2007) degrees in Computer Science and Information Technology from Riga Technical University. Her main research fields are IT project management, implementation and application of project management information systems, as well as project data analytics, big data and data science. She also works as a Team Lead at Accenture Latvia. She holds a PMP and CBAP certificates and was awarded the IPMA Outstanding Research Contribution of a Young Researcher 2013. She is a Member of PMI and Latvian National Project Management Association.
E-mail: Solvita.Berzisa@rtu.lv