DE GRUYTER OPEN
RIGA TECHNICAL UNIVERSITY
Information Technology and Management Science
ISSN 2255-9094 (online)
ISSN 2255-9086 (print)
December 2017, vol. 20, pp. 54–57
doi: 10.1515/itms-2017-0009
https://www.degruyter.com/view/j/itms

A Case Study: Software Defect Root Causes

Laila Bergmane[1], Jānis Grabis[2], Edžus Žeiris[3]
[1, 2] *Riga Technical University, Latvia,* [3] *ZZ Dats Ltd., Latvia*

***Abstract* – Software quality assurance to comply with user requirements enables software development companies to be competitive. Maintaining a high quality level requires continuous monitoring and development. If there are quality problems, the company's reputation is suffering and its costs increase because of investing in time and eliminating the consequences of the problems. The aim of the present article is to identify the most essential root causes of software defect. The e-service "Invoice Submission" of Riga City Municipality is used as an example. The results of the study can provide useful information for developing improvement activities for e-service higher quality. The analysis is based on the information that is available in the developer's user request database. The Ishikawa method is used to analyse the causes of defects.**

***Keywords* – Defect analysis, defect causes, software quality.**

I. Introduction

A human being can make an error, which produces a defect in a program code document. If a defect in the code is executed, the system may fail to function properly causing a failure. Defects in software, systems or documents may result in failures, but not all defects do so. Defects occur because human beings are fallible and because there is time pressure, complex code, complexity of infrastructure, changing technologies, and/or many system interactions [12].

Correction of defects is costly and the cost increases exponentially with every subsequent stage. They also directly affect software development cycle-time. Therefore, defect prevention not only enables one to reduce costs but also minimises the development time [2].

Development organisations that deliver software-based systems have to face serious problems on how to control the progress of test activities and quality of software products throughout the project life cycle in order to estimate test completion criteria, and if the project will end on time [10]. Testing activities play a major role in quality assurance and their non-compliance with the requirements is often the reason why users are dissatisfied with the product.

The quality of software needs to be assured through a proper development process. The development process must be improved on a regular basis according to the actual usage feedback. If a bug is in software, in particular, it is necessary to investigate a root cause of the bug in order to work out a proper measure to prevent it from recurring. Many reasons contribute towards software bugs in the project such as product, process and project related reasons [3]. Companies that are not good enough to resolve defect causes, risk with their reputation, loss of customer loyalty and cost cutting effects. To mitigate these risks, it is essential to perform improvement activities, which will improve the quality of the software. This requires a problem analysis of the causes of product defects.

A defect causal analysis has three key principles:

- reducing defects to improve quality;
- applying local expertise;
- focusing on systematic defects.

The first principle says that we can improve software quality by focusing on the prevention and early detection of defects. The second determines that the cause detection must involve a software development team that can explain why these defects have occured. The third principle says that, with relatively small investments, the focus on systemic defects can have a significant impact on quality [6].

To determine the context in this paper, the definition of the "defect" of the ISTQB term is used – a flaw in a component or system that can cause the component or system to fail to perform its required function, e.g., an incorrect statement or data definition. A defect, if encountered during execution, may cause a failure of the component or system [11]. A program is said to be buggy if that contains a large number of bugs, or bugs that seriously interfere with its functionality [3].

The quality requirements are determined by standards and internal quality procedures of companies [8]. Software quality must meet user requirements. There is no software which does not have a defect. Even in the case where no defects can be found in the software, this does not prove that they are not there. In the software development industry, both nationally and globally, the competitiveness of a company is closely related to its ability to develop high quality information technology solutions. Therefore, quality related issues are important for any software development company. Maintaining the quality level according to user requirements requires continuous management and preventive action.

E-service "Invoice Submission" is a service whose availability is disturbed due to existing defects. The quality problems indicate that there are systematic repetitions of software defects that significantly impact its full use for users. Misleading software adversely affects the company's reputation, and defect fixing and resources spent to fix these problems increase business costs, and instead of using the resources to develop new solutions, they are used to resolving existing software defects. The aim of this study is to investigate the root causes of defects. The e-service "Invoice Submission" has been used for a case study. User request database provides data for identifying the root causes. User requests in the database are classified information. The causes of the defects are given in a description way. Results of this paper can be used for continuous improvement activities in e-service quality improvement.

II. Related Research

There have been several studies which analysed root causes of defects in different systems using one or more data sources.

A study published in 2001 investigated 40 incident cases of web site functionality failures and found out that 80 % of all failures were software failures and human errors. A large number of failures occured during routine maintenance, software upgrades and system integration. The authors of the paper could not find out whether these failures wetr mainly due to system complexity, inadequate testing and/or poor understanding of system dependecies. They also indicated that other significant causes of software failure were system overload, resource exhaustion and complex fault recovery routines [5].

In a study about software development companies, in which 36 highly-qualified quality assurance testers and developers from open source projects were interviewed, it was concluded that the most common quality problems were due to the fact that the tester did not have enough information about the software to be tested. It was also mentioned that testing was rigorous, and no fair software quality assurance policy was available in written form [8].

In a study, using the defect classification approach, algorithm and functional type defects during the development process were found late – during system integration and testing. The mistakes were related to human factors – individual errors and lack of domain knowledge about a specific industry and system [1].

A case study about C compiler from the GNU Compiler Collection, which is an application consisting of over 300 000 lines of codes and can be divided based on functionality into 13 well-defined components, showed that a significant percentage of software failures were associated with changes that spread across the system, i.e., were due to nonlocalized faults. The authors of the study also analysed fligh software failures of NASA mission. The software includes multiple software applications, consisting of millions of lines of code in over 8000 files. The authors of the paper states that the most common sources of failures were requirements and coding faults, each contributing to about 33 percent of the failures. Requirement faults included incorrect, changed and missing requirements. The third largest fault type was related to data problems and it contributed to 14 percent of the failures. Design faults led to less than 6 percent of the failures. Additionally, 4 percent of failures were due to process or procedural issues, 2 percent – due to integration faults, and 1 percent – due to simulation or testing problems [9].

A study, which focused only on failures caused by defects in data-parallel programs, showed that most failures (84.5 %) were caused by defects in data processing rather than defects in code logic. The authors emphasised, "the tremendous data volume and various dynamic data sources make data processing error-prone". They also concluded that 22.5 % of failures are *table-level* and their major reasons were programmers' mistakes and frequent changes of data schema. There were also 62 % of r*ow-level* failures and most of them were caused by exceptional data. The authors concluded that programmers could not know all of exceptional data in advance [7].

Almost all of the studies under review [1], [8], [9] indicated that the revealed defects were related to testing problems. Other reasons were related with change management, lack of knowledge, requirements faults, data processing problems, and defects in code logic. Testing problems are typically related to testing process improvement activities. One of the most popular options how to improve this process at companies is to use one of the test maturity models, such as TPI Next, CMMI or TMMi, which helps companies evaluate the current testing situation and, based on the recommendations or guidelines suggested by the models, move towards improving the testing process at the company [14].

III. Research Methodology

The e-service "Invoice Submission" of Riga City Municipality was selected for the case study.

The e-service "Invoice Submission" is an online service that offers its users an electronic submission of invoices via the web service API, XML file uploads or manual invoice information entry. The invoice data validation is against the XSD scheme, which provides solutions for expanding or limiting input data at different levels.

The development of the e-service project started in 2011 and its development till today has been carried out in ten phases. The project team consists of project manager, tester, system analyst and programmer. The project manager is the only one who works on the project from its beginning, but other team members have changed several times.

Within the present research, several questions have been formulated:

- Why do the same defects repeat systematically?
- Does the technology used in the example solution affect the quality of e-service?
- What are the most common causes of defects in the production environment?

The data analysed were obtained from the supplier quality management information system. All defect requests received from users that were classified as a "defect" were selected from the database. The selected requests in the database were registered in the period of 5 April 2017–7 July 2017–. A total of 102 defect requests were received during this period. For analysis, the authors used the resolved defects and the information provided by the programmer and tester about the progress of defect handling and their causes. Each defect mentioned in the request was assigned to an apropriate category (Table I). IEEE Standard Software Anomaly was used to categorise the defects. They were classified by considering impact on requirement classes [4]. Functional defects were subdivided into the subcategories (categories 1–3). Thereafter, the number of defects in each category was determined.

TABLE I

DEFECT CATEGORIES

No.	Category	Description
1.	Development process	Actual or potential cause of failure is due to deficiencies in the requirements analysis, development, testing, implementation or maintenance process
2.	Integration with other systems	Actual or potential cause of failure is related to interoperability
3.	Data processing	Actual or potential cause of failure is any defect affecting data integrity
4.	Usability	Actual or potential cause of failure is related to usability (ease of use) requirements.
5.	Security	Actual or potential cause of failure is related to security requirements, such as those for authentication, authorisation, privacy/confidentiality, accountability (e.g., audit trail or event logging), etc.
6.	Performance	Actual or potential cause of failure is related to performance requirements (e.g., capacity, computational accuracy, response time, throughput, or availability).
7.	Serviceability	Actual or potential cause of failure is related to requirements for reliability, maintainability, or supportability (e.g., complex design, undocumented code, ambiguous or incomplete error logging, etc.).
8.	Other	Would not cause any of the effects above

To analyse the causes of software defects, the Ishikava method was used [13]. Based on the examples of method approach, four categories of causes were identified [6]:

- methods, which might be incomplete, ambiguous, wrong, or unenforced;
- tools and environment, which might be clumsy, unreliable, or defective;
- people, who might lack adequate training or understanding;
- input and requirements, which might be incomplete, ambiguos, or defective.

As a result of the analysis, the most important and most common causes were summarised and grouped by cause categories, depicting them in the form of Ishikawa diagram.

IV. RESULTS

In 2015, 17 defect requests were received from e-service users and this number increased to 64 in the subsequent year. Compared with 2015, it is at least three times more. The trend of 2017 in the first three months shows that the number of requests does not decrease significantly, but rather increases. Their number at the moment of the study has already reached 21 defect requests.

The number of defect requests obtained by classifying all defects by category is given in Table II.

TABLE II

TOTAL DEFECTS BY CATEGORY

No.	Category	Number of defects
1.	Development process	59
2.	Integration with other systems	10
3.	Data processing	20
4.	Usability	1
5.	Security	0
6.	Performance	1
7.	Serviceability	0
8.	Other	11

According to the results provided in Table II, the missed defects in the production environment are due to deficiencies in the development process. Among all user requests, in 74 cases of defect causes, it was mentioned that they had not been found during testing. Part of the defects could have been discovered earlier if the test method based on testing the characteristics had been used.

In some cases, the nature of the defects makes it clear that the tester lacks general knowledge about the e-service solution architecture and it is not enough to do integrity tests. In four cases, defects were reported in the e-service because the tester had not notified about new software defect removals that would require testing. Other five cases showed that defects occurred because functionality had been changed in another part of the system. Several defect requests showed that in many cases the tester and the system analyst had not been informed about changes implemented in functionality, thus creating situations where the described functional requirements were not actual in documents. All of these cases indicate problems in the change management process.

The second highest number of defects is related to data processing. These defects relate to generation of a PDF document that is affected by the degree of complexity of invoices. These problems cannot be completely avoided due to the technology used in the solution. Because the number of functionality defects is significantly higher than in other categories, this proportion indicates that the shortcomings of the development process and the problems with generating PDF documents most significantly affect the quality of the e-service. According to the defect rate, the third largest defect category is defects that have various other reasons, such as the temporary unavailability of a web service or a server. Some of the cases are affected by the human factor.

As a result of the analysis, the main causes of the defects are summarised in Fig. 1. It shows that the main causes of problems include categories “methods” and “human”. It demonstrates that there is an opportunity to improve software development process.

The study identified that systematic repetition of defects was due to the fact that the solution used restrictive technologies that could affect the quality of the service.

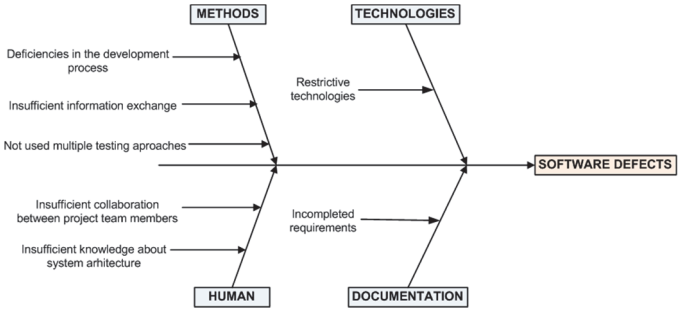

Fig. 1. Defect causes.

The results of the analysis show that functionality defects are the most common ones in the production environment, a total of 89 out of 102. The main causes of defects are related to problems in the development process, the technologies used in the solution, insufficient testing and lack of knowledge of the solution architecture.

V. Conclusion

The results of the study have confirmed the causes of the defects mentioned in the related studies; namely, the most common causes of defects are related to testing problems [9] and awareness of changes in software [8].

The main conclusions of the research are as follows:

- the example studied shows that the technology used in the solution may limit the choices of quality improvements, but does not prove that this is a common problem;
- one of the most common root causes of defects is related to deficiencies in the development process, which is also confirmed by the example under consideration;
- the used example shows that it is necessary to create new defect cause categories, which is useful for a systematic defect identification step in the defect cause analysis process;
- defect cause classification will help more precisly identify if the problems in the development process are related to design/analysis, coding, testing or infrastructure fields;
- in the example above, 71 % of all defects were not found during testing and this was mentioned in the related studies as one of the main causes of failures.

The limitation of the present study is related to the fact that the causes of the example software defects are identified based on the information provided in defect requests, the number of which may not be sufficient to ensure more objective determination of their causes.

Future research may be devoted to the implementation of new software defect cause classification at the company in order to support the defect cause analysis process.

References

[1] M. Leszak, D. E. Perry, and D. Stoll, "A Case Study in Root Cause Defect Analysis," *Proceedings of the 22nd international conference on Software engineering – ICSE '00*, pp. 433–434, Jun. 2000. https://doi.org/10.1145/337180.337232

[2] A. A. Shenvi, "Defect Prevention with Orthogonal Defect Classification," *Proceeding of the 2nd annual conference on India software engineering conference – ISEC '09*, p. 83, 2009. https://doi.org/10.1145/1506216.1506232

[3] V. Gupta, N. Ganeshan, and T. Singhal Kumar, "Determining the Root Causes of Various Software Bugs Through Software Metrics", *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, 11–13 March, New Delhi, p. 1212, 2015.

[4] "IEEE Standard Classification for Software Anomalies," *Revision of IEEE Std 1044-2009*, p. 8, 2010. https://doi.org/10.1109/ieeestd.2010.5399061

[5] S. Pertet and P. Narasimhan, *Causes of Failure in Web Applications.* Parallel Data Laboratory Carnegie Mellon University Pittsburgh, PA 15213-3890, CMU-PDL-05-109, pp. 3–8, Dec. 2005 [Online]. Aviable: http://www.cs.cmu.edu/~priya/PDL-CMU-05-109.pdf

[6] D. N. Card, "Learning from Our Mistakes with Defect Causal Analysis," *IEEE Software*, vol. 15, no. 1, pp. 56–63, 1998. https://doi.org/10.1109/52.646883

[7] S. Li, H. Zhou, T. Xiao, H. Lin, W. Lin, and T. Xie, "A characteristic study on failures of production distributed data-parallel programms," *Proceeding of the 2013 International Conference on Software Engineering*, p. 963–972, 2013. https://dl.acm.org/citation.cfm?id=2486921&CFID=1010944701&CFTOKEN=99591560

[8] N. Nuzhat, K. Aihab, and K. Ahmed, "Survey to Improve Software Quality Assurance in Developing Countries," *International Journal of Technology and Research, Islamabad 3.1 1–6*, pp. 3–5, 2015.

[9] M. Hamill and K. Goseva-Popstojanova, "Common Trends in Software Fault and Failure Data," *IEEE Transactions on Software Engineering*, vol. 35, no. 4, pp. 484–496, Jul. 2009. https://doi.org/10.1109/tse.2009.3

[10] N. Hrgarek, "Fabasoft Best Practices and Test Metrics Model," *Journal of Information and Organizational Sciences*, vol 31, no 1, Austria, pp. 75, 2007 [Online]. Available: http://jios.foi.hr/index.php/jios/article/view/31/29

[11] "ISTQB Glossary" [Online]. Available: http://glossary.istqb.org/search/defect

[12] ISTQB, "Why is Testing Necessary," *Certified Tester, Foundation Level Syllabus*, p. 11, 2012 [Online]. Available: http://www.istqb.org/downloads/send/2-foundation-level-documents/3-foundation-level-syllabus-2011.html4

[13] D. Dhandapani, "Applying the Fishbone diageam and Pareto principle to Domino," 2004 [Online]. Available: https://www.ibm.com/developerworks/lotus/library/fishbone/

[14] ISTQB, "Types of Process Improvement," *Certified Tester, Advanced Level Syllabus Test Manager*, p. 60, 2012 [Online]. Available: http://www.istqb.org/downloads/send/10-advanced-level-syllabus-2012/54-advanced-level-syllabus-2012-test-manager.html

Laila Bergmane is a student at the professional Master study programme "Information Technology" at the Institute of Information Technology of Riga Technical University (Latvia). She holds ISTQB certificate. This is her first publication. She also works as a Software Tester at ZZ Dats Ltd.
E-mail: Laila.Bergmane@edu.rtu.lv

Jānis Grabis holds a Doctoral degree and is a Professor at Riga Technical University (Latvia) as well as the Head of the Institute of Information Technology. His main research interests lie within the application of mathematical programming methods in information technology, enterprise applications and system integration. He has published more than 60 scientific papers, including a monograph on supply chain configuration. He has led a number of national projects and participated in five projects in collaboration with the University of Michigan-Dearborn (USA) and funded mainly by industrial partners, such as SAP America and Ford Motor Company.
E-mail: Grabis@rtu.lv

Edžus Žeiris holds a Doctoral degree and is a Deputy Director at the ZZ Dats Ltd. He holds CISM and PMP certificates. Research interests include design of e-services, security and evaluation of electronic services.
E-mail: Edzus.Zeiris@zzdats.lv