What Software Test Approaches, Methods, and Techniques are Actually Used in Software Industry?

Laura Strazdiņa,[1[0000-0002-3359-7099]], Vineta Arnicane [1[0000-0003-3942-9229]], Guntis Arnicans [1[0000-0002-8626-7595]], Jānis Bičevskis [1[0000-0001-5298-9859]], Juris Borzovs [1[0000-0001-7009-6384]], and Ivans Kuļešovs[2[0000-0002-7690-2807]]

[1] Faculty of Computing, University of Latvia, Raiņa bulvāris 19, LV-1586, Riga, Latvia
[2] SIA "C.T.Co", Meistaru iela 33, Valdlauči, Ķekavas novads, LV-1076, Riga, Latvia
laurastrazdina7@gmail.com, ivans.kulesovs@gmail.com
{vineta.arnicane,guntis.arnicans,janis.bicevskis,juris.borzovs}@lu.lv

Abstract. This paper presents the findings of, to the best of our knowledge, the first survey on software testing practices carried out in Latvian ICT industry. A total of 19 organizations participated in the survey, which was conducted in 2018. The survey focused on major aspects of software testing, namely testing approaches, strategies, methodologies, methods, and techniques. Based on the survey results, current practices in software testing are reported, as well as some observations and recommendations for the future of software testing in Latvia for industry and academia.

Keywords: Software testing, Software industry, Test approach, Test method, Test technique

1 Introduction

As Schaefer put it [1], systematic testing of software or systems can be learned, just like any engineering discipline. There are tester knowledge certification schemes: ISTQB, ISEB, GTB [2-4], there are books by Myers et al., Beizer, Kaner et al., to mention just few [5-7], and there are standards, e.g., ISTQB Glossary, BS 7925, IEEE 829, 1008, 1012, 29119, SWEBOK [8-11]. At least the books and most of the standards have been around for a long time, and many techniques are widely accepted. This means testing can actually be studied and then executed in some systematic way. For a tester or test engineer, there are two major activities: designing test cases, and executing test cases and observing and analyzing results. If the results are not like expected, deviations must be reported and followed up. Additionally, modern methods, like *exploratory testing* [12], *run-time verification* [13-14] include tasks like *automation* and *management of testing time* in the tester's task list. The normal way of doing this job is to learn some techniques, follow these techniques, execute the test, and conclude the work with a test report.

However, books and standards describe a lot of different ways (strategies, approaches, methodologies, methods, techniques) to perform testing [e.g., 15]. Are all

Lupeikiene A., Matulevičius R., Vasilecas O. (eds.):
Baltic DB&IS 2018 Joint Proceedings of the Conference Forum and Doctoral Consortium.
Copyright © 2018 for this paper by the papers' authors. Copying permitted for private and academic purposes.

these ways actually used in industry? If not, what are the most widely used?

There were several surveys to answer these questions.

A survey performed in 2004 in Australia by Ng, Murnane, Reed, Grant, Chen shows [16] that in general, test case derivation is reasonably widely used among the respondents. The survey results also reveal that deriving test cases from specifications (i.e., using black-box strategies) was likely to be more popular than deriving test cases from program codes (white-box strategies) in industry. However, there still exists a significant fraction of practitioners performing ad-hoc testing activities in Australia.

Scott, Zadirov, Feinberg, and Jayakody [17] in South Africa in 2004 found that software tests most widely performed by industry are: *unit testing, integration testing, acceptance testing, stress testing, load testing, performance testing, regression testing, usability testing, recovery testing, security testing, compatibility testing, beta testing*.

A survey of unit testing practices was done in Sweden in 2006 [18]. According to the author, Par Runeson, the survey revealed a consistent view of unit testing's scope, but participants did not agree on whether the test environment is an isolated harness or a partial software system. Furthermore, unit testing is a developer issue, both practically and strategically. Neither test management nor quality management seems to impact unit testing strategies or practices. Unit tests are structural, or white-box based, but developers rarely measure their completeness regarding structural coverage. Most of the companies surveyed desired unit test automation but had trouble spreading good practices across companies.

Itkonen, Mäntylä and Lassenius [19] in 2009 performed a rather limited exploratory study of manual testing practices. They identified 22 manual testing practices used by professionals. 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. The authors 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.

Among the findings of survey performed in 2009 in Canada [20] by Garousi and Varma were the following: (1) almost all companies perform unit and system testing, (2) automation of unit, integration and systems tests has increased sharply since 2004, (3) more organizations are using observations and expert opinion to conduct usability testing, (4) Junit and IBM Rational tools are the most widely used test tools.

Lee, Kang and Lee [21] in 2012 conducted a survey with a wide variety of companies and experts from Fortune 1000 companies that are involved in software testing in order to identify the current practices and opportunities for improvement of software testing methods and tools. The survey results revealed five important findings regarding the current practices of software testing methods and tools and opportunities for improvement: low usage rate of software testing methods and tools, difficulties due to a lack of software testing methods and tools, use of testing tools in a limited manner, demand for interoperability support between methods and tools of

software development and testing, and need for guidance to evaluate software testing methods and tools or to describe the capabilities of software testing methods and tools.

As far as we know the last and the most representative survey on usage of testing techniques and tools was performed by ISTQB [22]. According to this survey the most adopted test techniques are *use case testing* (70,8%), *exploratory testing* (66,3%), *checklist based testing* (64,1%), *boundary value analysis* (48,2%), *error guessing* (37%), *equivalence partition* (34%), *decision tables* (27,7%), *decision coverage* (21,5%), *state transition* (21,4%), *statement coverage* (18,2%), *pair-wise testing* (13,2%), *attacks* (10,4%), *classification tree* (7,1%). The most adopted non-functional testing activities include *testing of performance* (63%), *usability* (56,1%), *security* (38,5%), *reliability* (30,7%), *accessibility* (29,1%), *testability* (27,7%), *efficiency* (25,9%), *availability* (25,6%), *maintainability* (18,9%), *interoperability* (18,5%).

Although our survey comprises only major Latvian software developers, it differs in that is based on ISTQB Glossary [8]. The research questions are the following:

1) What testing approaches, methods and techniques mentioned in the Glossary are not used in practice and probably need not to be taught?
2) Is there any approach, method or technique used in practice that is not included in the Glossary?

The paper is structured as follows: Section 2 presents survey methodology. Section 3 presents survey results. Section 4 contains analysis and summary of survey findings, Section 5 – conclusions and future work.

2 Survey Methodology

The University of Latvia conducted a survey on software testing in Latvia.

2.1 Survey Objectives

The primary objective of the survey was to determine testing methods, techniques, approaches used by software testers and developers in Latvia when they carry out software testing activities. The aim was to find out the best testing practices currently used in the IT industry of Latvia.

The second objective was to determine whether all testing methods, techniques, approaches included in ISTQB glossary are well known by testers and developers.

The third objective was to explore if the usage of testing methods, techniques, approaches depends on the size of the company in the means of the count of developers and testers.

2.2 Survey Description

The survey targeted senior employees involved with testing or development in software development companies or departments of software development in enterprises with another kind of main business.

There were three parts in the survey. The first part was an introductory section where the aims of survey and way how to fill it was described.

The second part of the survey comprised questions about the size company count of employees working in software development and how many of them are the software testers.

The third part was the main part of the survey – 153 alphabetically ordered testing methods, techniques, and approaches (further in the paper called as methods) based on the software testing glossary of ISTQB.

2.3 Survey Method

The respondents of the survey were invited to complete the survey online at survey website (see the list of methods of the survey in the Appendix).

Each question had predefined answers: 1) *My company frequently uses this testing* method; 2) My *company rarely uses this testing method*; 3) *My company does not use it*; 4) *I don't know this method*: 5) *Other…*

The open-type answer '*Other…*' was used to allow respondent give some specific answer. Respondents were informed that survey terms are taken from ISTQB glossary so they could use the ISTQB home page to get acquainted with the explanations of terms.

It was presumed that fulfillment of survey would take approximately 20 minutes.

Confidentiality and privacy were assured to all respondents and the organization that they represented.

2.4 Sample Selection

Our survey was targeted the employees at the organizational level for software development companies and at the departmental level if there is a department for software development in the enterprise and the software development is not its main business. The first preference was test managers; the second was software project managers who know the overall situation in the software testing in the organization.

3 Survey Results

As a result, a total of 19 companies participated in the survey, representing more than 700 software developers including more than 380 testers. The survey results show which test approaches, methods, and techniques are used in the IT industry of Latvia.

3.1 Organization Information

All of the 19 organizations responded to our survey are organizations that mainly focus on software development or have a software development department and are located in Latvia. 53% of the respondents responded that their organization has more than 50 software developers employed. In one of the surveyed organizations, there are no testers employed (see Fig. 1).

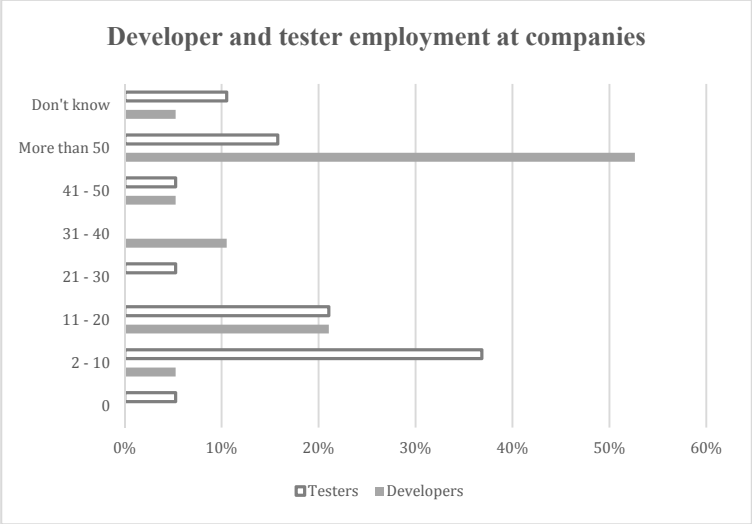

Fig. 1. Developer and tester employment at companies

3.2 Software Approaches, Strategies, Methodologies, Methods, and Techniques

According to the survey results most frequently used testing methods are *functional testing* (95%), *interface testing* (89%), *regression testing* (89%), *acceptance testing* (84%), *verification* (84%), *functionality testing* (84%) and *black-box testing* (84%). None of the respondents responded that they do not use *functional testing*, *interface testing*, *acceptance testing*, and *functionality testing*. However, 5% of participants responded that they do not use *regression testing* and *verification*; 11% of participants responded that they do not use *black-box testing* (Table 1).

Table 1. Most frequently used testing methods

Testing method	Use frequently	Use rarely	Do not use	Don't know	Other
Functional testing	**95**%	5%	0%	0%	0%
Interface testing	**89**%	11%	0%	0%	0%
Regression testing	**89**%	0%	**5**%	5%	5%
Acceptance testing	**84**%	11%	0%	0%	5%
Verification	**84**%	11%	**5**%	0%	5%
Functionality testing	**84**%	5%	0%	11%	0%
Black-box testing	**84**%	5%	**11**%	0%	11%

From the 19 surveyed organizations 63% responded that they do not use *fault seeding*. Other least used testing methods across the surveyed organizations are *outsourced testing* (58%), *malware scanning* and *mutation testing* (53%) (Table 2).

Table 2. Least used testing methods

Testing method	Use frequently	Use rarely	Do not use	Don't know	Other
Fault seeding	11%	16%	**63**%	5%	68%
Outsourced testing	5%	26%	**58**%	11%	58%
Malware scanning	11%	16%	**53**%	11%	63%
Mutation testing	21%	11%	**53**%	32%	37%

The results of the survey may not be completely precise since there are a few testing methods which quite a lot of respondents did not know. The testing methods that were the most unknown across respondents are *LCSAJ testing* (74% of respondents responded that they do not know what is *LCSAJ testing*), *PRISMA* (63%), *N-wise testing* (58%), *N-switch testing* (58%), *Wideband Delphi* (58%), *orthogonal array testing* (58%), *neighborhood integration testing* (53%) (Table3).

Table 3. The most unknown testing methods

Testing method	Use frequently	Use rarely	Do not use	Don't know	Other
LCSAJ testing	0%	11%	16%	**74**%	16%
PRISMA	0%	21%	16%	**63**%	16%
N-wise testing	0%	5%	37%	**58**%	37%
N-switch testing	0%	11%	32%	**58**%	32%
Wideband Delphi	0%	11%	32%	**58**%	32%
Orthogonal array testing	5%	5%	32%	**58**%	32%
Neighborhood integration testing	11%	21%	16%	**53**%	16%

Additionally to the mentioned testing methods in the survey one of the respondents mentioned that *crowdsourcing*, *in-sprint testing*, *feature/epic acceptance testing* are used in the represented organization.

There were 153 testing methods, techniques, and approaches questioned in the survey (see the list of them in the Appendix). 67 methods were used by at least 67% of the respondent organizations, additional 39 methods were used by 50%-67% of respondents. 16 methods were used by less than 33% of respondent organizations.

4 Analysis and Summary of Survey Findings

The survey results show that the testing method currently used the most in the IT industry of Latvia is *functional testing* while only 52.63% of respondents responded that they frequently use *non-functional testing*. Non-functional testing has some benefits, for example, evaluation of the overall performance of the system and whether the system's performance is as expected under normal and expected conditions.

The least used testing method is *fault seeding* (63% responded that they do not use it). Also only 58% of respondents responded that they do not use *outsourced testing*. External testing may be more effective sometimes since insourced testing's lack of objectivity often limits their effectiveness.

As the results show the least known testing method currently in the IT industry of Latvia is *LCSAJ testing*. 8 from all of the mentioned testing methods in the survey were responded to be unknown by more than 50% of respondents. Test managers should be informed about as much as possible testing method options to be able to plan the testing process for the system to achieve the highest quality.

When comparing these survey results with the before mentioned survey performed by ISTQB [22]. *Use case testing* is adapted 8,2% more, *exploratory testing* 23,3% less, *checklist based testing* 0,1% less, *boundary value analysis* 8,8% more, *error guessing* 1% less, *equivalence partition* 5% less, *decision tables* 6,7% less, *state transition* 7,4% less, *pair-wise testing* 0,8% more. *Exploratory testing* is important since it gives the opportunity to use the system more similar to how the end user would.

5 Conclusions and Future Work

In this paper, we presented and analyzed the findings of our preliminary software testing survey on software testing practices carried out in Latvian ICT industry, which was conducted in 2018. Although, only 14 organizations participated in the survey, the results of the survey give enough information to make conclusions.

As the next stage of the survey, we plan to carry out it in the Baltic States to find out the best testing practices currently used in the IT industry of Latvia, Lithuania, and Estonia. We would also like to compare the data from the Latvian ICT industry to that obtained from other Baltic States countries in order to see which testing methods are used more in which country.

Acknowledgements

The work described in this paper was supported by the project no. AAP2016/B032 "Innovative information technologies" at the University of Latvia. We are grateful to all respondents of the survey. Without their efforts and enthusiasms, this survey could never be successful. Finally, any errors or omissions are the faults of the authors.

Appendix

The list of testing methods included in questionnaire:

1. Acceptance testing
2. Accessibility testing
3. Ad hoc testing
4. Agile testing
5. Alpha testing
6. Analytical testing
7. API testing
8. Attack-based testing
9. Benchmark test
10. Beta testing
11. Big-bang testing
12. Black-box testing
13. Bottom-up testing
14. Boundary value analysis
15. Branch testing
16. Build verification test (BVT)
17. Business process-based testing
18. Capture/playback tool
19. Checklist-based testing
20. CLI testing
21. Combinatorial testing
22. Compliance testing
23. Component integration testing
24. Component testing
25. Concurrency testing
26. Condition testing
27. Confirmation testing
28. Consultative testing
29. Control flow testing
30. Conversion testing
31. Data flow testing
32. Data-driven testing
33. Database integrity testing
34. Decision condition testing
35. Decision table testing
36. Decision testing
37. Defect-based test design technique
38. Design-based testing
39. Desk checking
40. Development testing
41. Documentation testing
42. Dynamic testing
43. Efficiency testing
44. Elementary comparison testing
45. Equivalence partitioning
46. Error guessing
47. Experience-based testing
48. Exploratory testing
49. Factory acceptance testing
50. Failover testing
51. Fault injection
52. Fault seeding
53. Fault Tree Analysis (FTA)
54. Formal review
55. Functional testing
56. Functionality testing
57. Fuzz testing
58. GUI testing
59. Hardware-software integration testing
60. Hyperlink test tool
61. Incident management tool
62. Incident report
63. Incremental testing
64. Informal review
65. Insourced testing
66. Inspection
67. Installability testing
68. Integration testing
69. Interface testing
70. Interoperability testing
71. Invalid testing
72. Isolation testing
73. Keyword-driven testing
74. LCSAJ testing
75. Load testing
76. Maintainability testing
77. Maintenance testing
78. Malware scanning
79. Management review
80. Methodical testing
81. Model-based testing (MBT)
82. Monkey testing
83. Multiple condition testing
84. Mutation testing
85. N-switch testing
86. N-wise testing

87. Negative testing
88. Neighborhood integration testing
89. Non-functional testing
90. Operational acceptance testing
91. Operational profile testing
92. Operational testing
93. Orthogonal array testing
94. Outsourced testing
95. Pair testing
96. Pairwise integration testing
97. Pairwise testing
98. Path testing
99. Peer review
100. Penetration testing
101. Performance testing
102. Portability testing
103. PRISMA
104. Procedure testing
105. Process cycle test
106. Process-compliant testing
107. Random testing
108. Reactive testing
109. Recoverability testing
110. Regression testing
111. Regression-averse testing
112. Reliability testing
113. Requirements-based testing
114. Resource utilization testing
115. Risk-based testing
116. Robustness testing
117. Safety testing
118. Scalability testing
119. Scripted testing
120. Security testing
121. Session-based testing
122. Simulator
123. Site acceptance testing
124. Smoke test
125. Software Usability Measurement Inventory
126. Standard-compliant testing
127. State transition testing
128. Statement testing
129. Static testing
130. Statistical testing
131. Stress testing
132. Stress testing tool
133. Suitability testing
134. Syntax testing
135. System integration testing.
136. System testing
137. Technical review
138. Test harness
139. Test-driven development (TDD)
140. Thread testing
141. Top-down testing
142. Usability testing
143. Use case testing
144. User acceptance testing
145. User story testing
146. Validation
147. Verification
148. Volume testing
149. Vulnerability scanner
150. Walkthrough
151. Website Analysis and Measurement Inventory
152. White-box testing
153. Wideband Delphi

References

1. Schaefer, H.: What a tester should know, even after midnight, http://www.softwaretesting.no/testing/testermidnighteng.pdf, last accessed 2018/03/16.
2. ISTQB: International Software Testing Qualifications Board. www.istqb.org, last accessed 2018/03/16
3. ISEB: Information Systems Examinations Board of British Computer Society. https://certifications.bcs.org/, last accessed 2018/03/16
4. GTB: German Testing Board: www.german-testing-board.info, last accessed 2018/03/16
5. Myers, G.J., Badget, T., Sandler, C.: The Art of Software Testing. 3rd ed. John Wiley & Sons, Inc., Hoboken, New Jersey, USA (2012)

6. Beizer, B.: Black-box testing: techniques for functional testing of software and systems. John Wiley & Sons, Inc. New York, USA (1995)
7. Caner, C., Bach, J., Pettichord, B.: Lessons Learned in Software Testing: A Context-Driven Approach. John Wiley & Sons, Inc., Hoboken, New Jersey, USA (2008)
8. Standard Glossary of Terms used in Software Testing, Version 3.1., All Terms, International Software Testing Qualifications Board, https://www.istqb.org/downloads/send/20-istqb-glossary/186-glossary-all-terms.html , last accessed 2018/03/13
9. BS 7925-2-Standard for Software Component Testing. British Computer Society (1998).
10. IEEE Standards: www.ieee.org, last accessed 2018/03/16
11. Bourque, P., Fairley, R.E.: Guide to the Software Engineering Body of Knowledge (SWEBOK®: Version 3.0. IEEE Computer Society Press, Los Alamitos, CA, USA (2014).
12. Bach, J.: Exploratory Testing Explained. https://people.eecs.ku.edu/~hossein/Teaching/Fa07/814/Resources/exploratory-testing.pdf , last accessed 2018/03/16
13. Leucker, M., Schallhart, C.: A brief account of runtime verification. The Journal of Logic and Algebraic Programming 5, 293-303 (2009)
14. Bicevskis, J., Bicevska, Z., Rauhvargers, K., Diebelis, E., Oditis, I., Borzovs, J.: A practitioner's approach to achieve autonomic computing goals. Baltic Journal of Modern Computing 4, 273-293 (2015)
15. Kuļešovs, I., Arnicāne, V., Arnicāns, G., Borzovs, J. Inventory of testing ideas and structuring of testing terms. Baltic Journal of Modern Computing 3-4(1), 210-227 (2013)
16. Ng, S.P., Murnane, T., Reed K., Grant D., Chen T.Y.: A preliminary survey on software testing practices in Australia. In: Proceedings of the 2004 Australian Software Engineering Conference (ASWEC'04), pp. 116-125. IEEE Explore Digital Library (2004)
17. Scott, E., Zadirov, A., Feinberg, S., Jayakody, R.: The alignment of software testing skills of IS students with industry practices – A South African Perspective. Journal of Information Technology Education 3, 161-172 (2004)
18. Runeson, P.: A survey of unite testing practices. IEEE Software 4 (23), 22-29 (2006)
19. Itkonen, J., Mäntylä, M.V., Lassenius, C.: 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, pp. 494-497. IEEE, Lake Buena Vista, Florida, USA (2009)
20. Garousi, V., Varma, T.: A replicated survey of software testing practices in the Canadian Province of Alberta: What has changed from 2004 to 2009? The Journal of Systems and Software 83, 2251-2262 (2010)
21. Lee, J., Kang S., Lee, D.: A Survey on software testing practices. IET Software, June, 1-14 (2012)
22. ISTQB® Worldwide Software Testing Practices Report 2015-2016. https://www.istqb.org/documents/ISTQB_Worldwide_Software_Testing_Practices_Report.pdf , last accessed 2018/03/24