ISSN 1330-3651 (Print), ISSN 1848-6339 (Online)

https://doi.org/10.17559/TV-20180713101347
Original scientific paper

An Empirical Investigation of Software Testing Methods and Techniques in the Province of Vojvodina

Vuk VUKOVIC, Jovica DJURKOVIC, Marton SAKAL, Lazar RAKOVIC

Abstract: A high-quality test design is a conditio sine qua non of successful software testing process, and its effectiveness depends, among other things, on the choice and proper use of appropriate methods and relevant software testing techniques. The main goal of this study was to provide insight into the use of current methods and relevant software testing techniques used in the test design phase of software testing process in software companies in the Province of Vojvodina. The empirical study was conducted by a survey research strategy in twenty-four software organisations. Eighty-three respondents took part in the survey. Descriptive analysis, correlation analysis, hierarchical cluster analysis, the multidimensional scaling, binomial test and Cohran's *Q* test were used for analyzing gathered quantitative data. The survey results have shown that respondents use to a significant extent the techniques belonging to ISO/IEC/IEEE 29119 testing standard. Comparison of the gathered data with individual results of similar studies conducted in Canada, Australia and Turkey has shown similarities between them and companies in the Province of Vojvodina. The findings of this study present empirically verified recommendations for testing design phase realization in the form of least and most used software testing methods and techniques, their benefits, limitations and details in application, similarities between software testing techniques, software testing techniques clusters and the probability of use of individual techniques.

Keywords: software testing; survey; test design; testing methods and techniques

1 INTRODUCTION

Living in the era of pervasive computing, we become conscious of ubiquitous significance of software products not only in the case of its total absence, as foreseen by Beluzzo two decades ago [1], but also when software fails in "making life and business more comfortable and efficient" due to defects [1].

For a considerable amount of time, software defects have not been the exclusive topic of highly specialized, professional IT journals and web portals; their being practically daily phenomenon is (also) testified to in general daily press and portals of general subject area, writing about this phenomenon from the viewpoint of everyman customer.

The above claim can be supported by several illustrative, randomly picked examples of globally known cases of software defects occurring in the past few years: an online retailer offered items at the price of 1 penny [2], and an airline even offered free tickets [3]. A financial services company lost 440 million dollars within 30 minutes due to a software defect [4]. ATMs were out of order, leaving clients unable to withdraw their own money [5], whereas in other cases they allowed cash withdrawals at amounts higher than that available on the account [6]. Divorced couples were brought into a situation to "renegotiate the terms of their separation" [7]. Emergency services were unavailable to more than 11 million people for 6 hours [8], more than 600 suspected cases of child abuse were not reported to the police for a year and a half [9], about 3200 inmates were released from prisons earlier, etc. [10]. One can also deem as usual the news that software defects caused delay in the release of operative systems and software products [11], recall of cars [12], problems with consumer electronics [13], and issues with social networks [14].

Moreover, repercussions of software defects are not negligible, when viewed quantified, globally, as shown by the results of Cambridge University research [15]: developers spend a half of programming time on locating and correcting bugs, which, taking into consideration wages and overhead costs, results in a global cost of debugging software at the amount of $312bn per year. As stated in the above mentioned research, the stated amount is not finite; it should be increased by the amount of the arising opportunity costs. Software defects are a common cause of brand damage [16], with potentially detrimental impact not only on the reputation of the software producing organisations, but also those who use them.

The above stated data clearly point to the imperative of efficient integration of a modern testing process into software development. It is a life-cycle approach which includes different phases. An analysis identified four phases of the software testing lifecycle: test planning, test design, test execution and test evaluation [17]. These phases (sub-processes) of the testing process are also present in referent software testing models such as ISO/IEC/IEEE 29119, TMMi and TMap Next [18-20].

The subject of interest of this study is the test design sub-process. It presents the act of creating and writing test suites (collection of test cases) [19], requiring knowledge of testing techniques. A high-quality test design is a condition sine qua non of successful software testing process [21], and its effectiveness depends, among other things, on the choice and proper use of appropriate methods and relevant testing techniques.

The purpose of the study was to identify the most frequently used software testing methods and techniques in software organisations on the territory of the Province of Vojvodina, to note their specific features in implementation, and compare the research results with the results of similar studies conducted in other countries.

The paper is structured as follows: after a summary review of related work in Section 2, the objectives, design and realization of the empirical research are described in Section 3. The results are presented in Section 4, and the discussion in Section 5. Conclusions and directions of future research are given in the final, 6th section of the paper.

2 RELATED WORK

This study was substantially influenced by the research results which, among others, included software testing methods and techniques [22-26]. Their summary overview is given in the section below.

Ng et al. [22] conducted a survey in 65 organisations in Australia, encompassing five aspects of software testing, including the aspect of testing methodologies and techniques. The survey results point to the fact that 64.4% of organisation used at least one structural testing methodology in three years. As regards test case selection, black box testing techniques (especially boundary value analysis and random testing) hold sway over white box. Only three organisations use the mutation analysis testing technique. Static analysis (document and code inspection) in comparison with dynamic analysis (code walkthroughs) has a slight advantage (29 organisations compared to 22). 38.5% of organisations generate 80% of test cases based on specification, whereas 26.2% organisations generate between 60% and 79% of test cases based on specification. 33.8% organisations found that between 40% and 59% of software defects are related to errors in specification.

Starting from and slightly expanding the questionnaire of Geras et al. [23], which was used in a research conducted in the Province of Alberta, in their survey that encompassed a broader geographic area, Garousi & Varma [24] pointed to changes in the domain of software testing over a period of five years. The study indicated that the largest number of organisations (with a minor decrease of 5% in relation to 2004) still relies solely on software testers' skills and intuition. In relation to 2004, an increasing use is recorded for Boundary Values Analysis, States, Equivalence Classes Partitioning, Control Flow Graphs and Cause Effect Graphs. The most used testing techniques for prevention of defects in software are Reuse and Informal Inspection, although their use records a fall in relation to 2004.

Garousi & Zhi [25] devised a survey with the aim to identify software testing trends, and also to provide a view of testing techniques, tools and metrics used by professional software testers. When it comes to results about the use of testing techniques in the conducted study, a worrying fact is that most respondents do not use any of the testing techniques for generating test cases (black box or white box). Testing techniques belonging to black box testing method are more popular among respondents than testing techniques belonging to white box testing method. Boundary Value Analysis singled itself out as the most used testing technique, while somewhat fewer respondents opted for Equivalence Class Partitioning. The positive result of this survey is that a significant number of respondents use model-based testing (based on UML models). The respondents also use random testing and user-story based acceptance testing. Another surprising result of the study is a very modest use of explorative testing by respondents which is represented in agile software development processes. Mutation testing is also used very modestly.

Garousi et al. [26] conducted a survey aimed at providing a view of software engineering in Turkish software industry. Consequently, testing was included in this research, and testing techniques within it. According to survey results, the use of four testing techniques - Boundary Value Analysis, Equivalence Class Partitioning, Model Based Testing, and Source Code Analysis - is dominant in relation to other testing techniques, ranging between 22% and 33%.

3 RESEARCH DESIGN

The research goal and research questions were defined in section 3.1. The following section, 3.2, presents research strategies and data gathering instruments (questionnaire and interview scheme). Relevant data about the sample is detailed in Section 3.3, while Section 3.4 describes how the research was realized and what procedures were used for data analysis.

3.1 Research Goal and Research Questions

The goal of this study was to provide the latest insight into the use of current software testing methods and techniques in software organisations. In relation to the set goal, the following research questions (RQ) were defined:

- RQ1. What are the most and least used software testing methods?
- RQ2. What are the most and least used software testing techniques?
- RQ3. Are software testing techniques defined by a model or testing standard used by the organisation?
- RQ4. What are the specific features (details) of the application of identified most used software testing techniques?

Each of the research questions defined the content of questionnaire questions and interview schemes presented in the following section.

3.2 Research Strategies and Data Gathering Instruments

Empirical research consists of two components, quantitative (survey) and qualitative (interview). To provide preconditions for survey that should enable an insight into the use of actual software testing techniques in practice, we consulted studies whose results are presented in the Related work section [22-26]. Having overviewed the questions found in the questionnaires of the above mentioned studies and study results, we developed a questionnaire that we used in our survey. In addition to these studies, when developing the questionnaire, we also consulted the ISO/IEC/IEEE 29119 testing standard, whose fourth section contains recommended testing techniques for generating test cases [27]. It is one of the leading software testing standards, whose development included 27 countries, and was identified in the results of the systematic literature review conducted by Garcia et al. [28]. Most of the testing techniques from ISO/IEC/IEEE 29119 were included in the questionnaire, but we deliberately omitted a number of testing techniques from the questionnaire. In this manner, we wanted to encourage respondents to state the testing techniques that they use in practice, which are, however, not stated in the questionnaire. The content of the final version of the questionnaire was tested by two test managers with rich experience so as to remove dilemmas, and harmonize the wording of the questionnaire with the terminology used in practice. The questionnaire had the structure shown in Tab. 1. It also links the questionnaire questions with the previously listed research questions.

Having identified the most used software testing techniques, the qualitative component of research was performed, with the intent of answering the RQ4. This implied conducting interviews with experts in the domain of software testing according to the scheme shown in Tab. 2.

Table 1 Questionnaire structure

Aspect (RQ)	Questions (Metrics)
Organisations and respodents	1. Designation and number of employees in organisation 2. Occupation (diploma) 3. Position
What are the most and least used software testing methods? (RQ1)	4. Which testing methods do you use in the software product testing process: black box, white box or both methods?
What are the most and least used software testing techniques? (RQ2)	5. Which testing techniques do you use in the software product testing process (Peer reviews, Equivalence Class Partitioning, Boundary Value Analysis, Combinatorial Test Techniques, Cause-Effect Graphing (with Decision Table Testing), Statement Coverage, Branch Coverage, Condition Coverage, Loop Coverage, Control Flow Graphs, Code Complexity Analysis)? 6. Please state the testing techniques that you use, but are not listed above 7. Software product testing in our organisation is performed during the exploratory testing.
Are software testing techniques defined by a model or testing standard used by the organisation? (RQ3)	8. Are software testing methods and techniques used in your organisation's software projects defined by a model or testing standard used by your organisation?

Table 2 Interview scheme

Aspect (RQ)	Questions
What are the specific features (details) of the application of identified most used software testing techniques? (RQ4)	1. What are the benefits of identified most used testing techniques (this question referred to the 7 identified testing techniques from RQ2)? 2. What are the limitations of identified most used testing techniques (this question referred to the 7 identified testing techniques from RQ2)? 3. Details of application of the most used testing techniques (this question referred to the 7 identified testing techniques from RQ2)? 4. In which cases is a particular testing technique better than another? 5. Is the application of identified software testing techniques conditioned by the type of software product?

3.3 Respondent Population and Sample Size

The survey research strategy was applied on a convenience sample of 24 organisations (83 respondents) of the basic set of organisations producing software in Province of Vojvodina. The required sample sizes for the survey were calculated by a priori analysis based on the type of procedure, set-size effects, probability α and the set power of the statistical test $(1 - \beta)$ [29, 30]. Two-tailed hypothesis testing in the correlation analysis: ρ $H0 = 0$; ρ $H1 = 0.3$; $\alpha = 0.05$; power of test $(1 - \beta) = 0.80$ [30]; required sample size (in number of items) $n = 84$.

Due to incomplete response in the e-survey, the actual sample size was $n = 83$. Post hoc analysis was used to calculate that for the two-tailed hypothesis testing in the correlation analysis ρ $H0 = 0$; ρ $H1 = 0.3$; $\alpha = 0.05$; where the actual sample size $n = 83$ and the power of test $(1 - \beta) = 0.795$, therefore, slightly less than the originally required value $(1 - \beta) = 0.80$. The sample size $n = 83$ was sufficient for applying other statistical tests in the quantitative analysis of data obtained by this survey.

The qualitative component of research was conducted on a convenience sample of 12 experts possessing the required types of knowledge, skill, and information from the problem domain.

3.4 Research Execution and Data Analysis Procedures

The research instrument applied on the convenience sample of organisations was the questionnaire which was distributed to the selected organisations through the web form created in Google forms. Having completed this research component, the respondents' answers were taken over in the Google spreadsheet format and subjected to the analysis of quantitative data. The following procedures were used for analyzing quantitative data: descriptive analysis data, correlation analysis, hierarchical cluster analysis, the multidimensional scaling, binomial test and Cochran's Q test. All these procedures were executed in MS Excel and SPSS Statistics program packages.

Having completed the analysis of the quantitative data, the most used software testing techniques were identified, whose specific features in application (benefits, limitations, details about the usage) were the subject of the qualitative research component. By means of recorded face to face interviews, data were gathered whose transcripts were made in text processor, after which they were sent to experts for verification and confirmation. The transcribed data were then inserted into NVivo software, where the thematic analysis of their content was performed.

4 RESULTS AND FINDINGS

This section presents the results of the empirical research. Survey results are presented in the same order as in Tab. 1, in sections 4.1 through 4.4, while the results of the qualitative research are presented in section 4.5.

4.1 Results on Organisations and Respondents

The number of employees in organisations included in the survey sample is presented in Fig. 1. The organisation sample is heterogeneous by the criterion of organisation size. The parameter of number of employees was used for determining the size of organisation, whereas the annual capital turnover was omitted due to data confidentiality. With the exception of one, the sample comprises organisations which, according to European Commission [31], belong to categories of micro, small, and medium-sized organisations.

To establish the respondents' level of education, data was collected on the highest academic degree (occupation) that they possess. As Fig. 2 shows, the largest number of respondents have a MSc degree (65.07%), where as

33.73% respondents have a BSc degree and one respondent PhD degree.

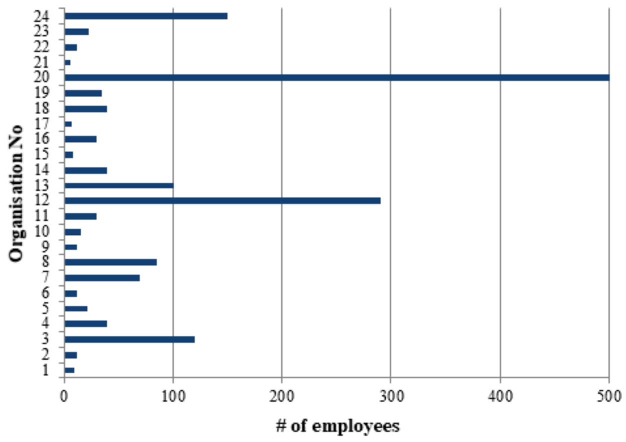

Figure 1 Number of employees in organisations comprising the sample

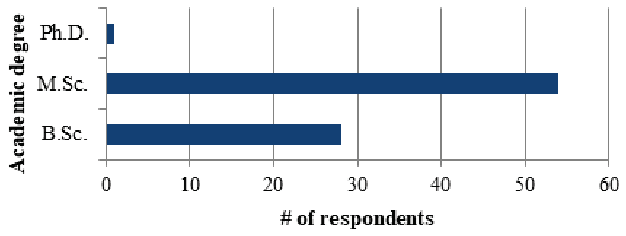

Figure 2 The highest degrees of respondents in organisations

The organisational positions of respondents who participated in this research are presented in Fig. 3.

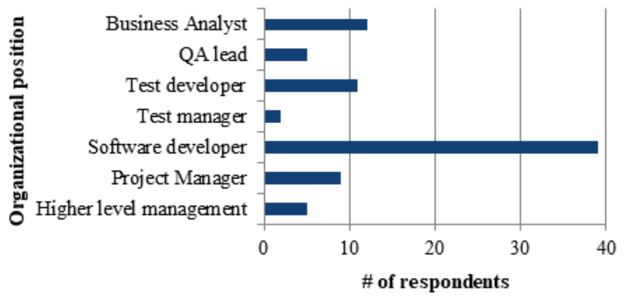

Figure 3 Respondents' positions in organisations

When distributing the questionnaire, we asked the HR departments of the surveyed organisations to forward the questionnaires to persons who are currently working, or were working on software testing in the past, so as to provide high-quality and relevant data. Fig. 3 shows that the highest percentage of respondents are employed as software developers (46.98%), a total of 21.68% on the positions of QA lead, test developer and test manager, a total of 16.86% on the position of Project Manager and Higher level management, whilst 14.45% respondents work on the position of Business.

4.2 Use of Software Testing Methods

The data on distribution of the frequency of application of software testing methods are shown in Fig. 4.

The highest percentage of respondents, 65.1%, use both tests methods (black box and white box) when performing the software product testing process, while 21.7% respondents use only the black box method, and 13.3% only the white box method.

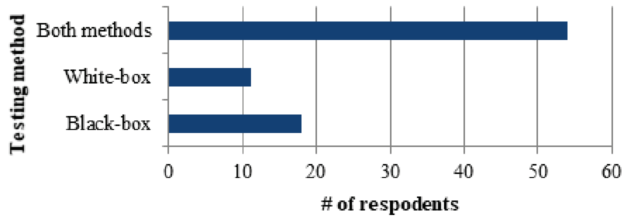

Figure 4 Frequency of the application of software testing methods

The data on distribution of the frequency of application of software testing methods according to organisation size are shown in Fig. 5.

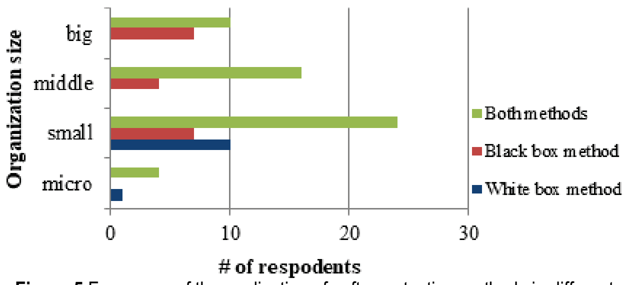

Figure 5 Frequency of the application of software testing methods in different organisation sizes

4.3 Use of Software Testing Techniques

Data on the frequency of the use of individual software testing techniques in the surveyed software organisations are shown in Fig. 6. The respondents were able to use the questionnaire to select the offered testing techniques that they use when testing software product (question 5), and list the testing techniques that they use, but which were not defined in the questionnaire (question 6).

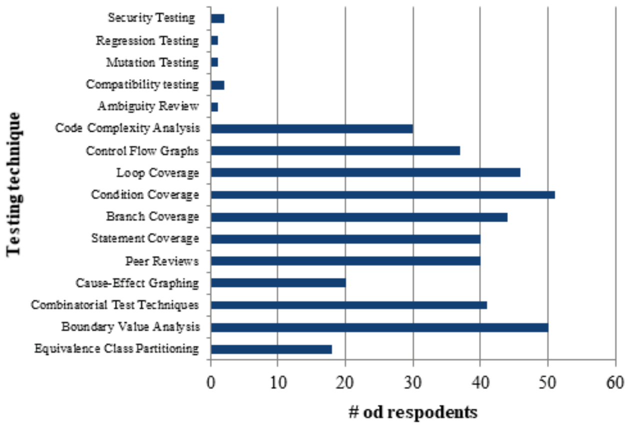

Figure 6 Frequency of the use of individual testing techniques

Boundary Value Analysis (BVA) with 60.2% and Condition Coverage (CC) with 61.4% singled themselves out as the most used testing techniques. Also, a significant percentage of the frequency of use is found with the techniques Combinatorial Test Techniques (CTT), Peer Reviews (PR), Statement Coverage (SC), Branch Coverage (BC) and Loop Coverage (LC). The techniques of Equivalence Class Partitioning (ECP), Cause-Effect Graphing (CEG), Control Flow Graphs (CFG) and Code

Complexity Analysis (CCA) have a lower percentage of use frequency in surveyed organisations.

Due to minor use and unfamiliarity of testing terminology found in some respondents, the testing types and techniques Ambiguity Review (AR), Compatibility Testing (CT), Mutation Testing (MT), Regression Testing (RT) and Security Testing (ST) were excluded from further quantitative analyses. Actually, compatibility and security testing represent the types of software testing that are not the subject of research in this study.

Data on the frequency of the application of testing techniques without five omitted testing types and techniques are shown in Tab. 3.

Table 3 Data on the application of testing techniques

Technique name	Responses		% of cases
	N	%	
Equivalence Class Partitioning	18	4.3%	21.7%
Boundary Value Analysis	50	12.0%	60.2%
Combinatorial Test Techniques	41	9.8%	49.4%
Cause-Effect Graphing	20	4.8%	24.1%
Peer Reviews	40	9.6%	48.2%
Statement Coverage	40	9.6%	48.2%
Branch Coverage	44	10.6%	53.0%
Condition Coverage	51	12.2%	61.4%
Loop Coverage	46	11.0%	55.4%
Control Flow Graphs	37	8.9%	44.6%
Code Complexity Analysis	30	7.2%	36.1%
Total	417	100.0%	502.4%

The following section of the paper gives a tabular overview of the distribution of frequency of use of individual software testing techniques. Data with the highest and lowest frequency of technique application for testing methods, both black box and white box, are shown.

Table 4 Distribution of use frequency of Boundary Value Analysis

	Frequency	Percent	Cumulative percent
No	33	39.8	39.8
Yes	50	60.2	100.0
Total	83	100.0	

Table 5 Distribution of use frequency of Equivalence Class Partitioning

	Frequency	Percent	Cumulative percent
No	65	78.3	78.3
Yes	18	21.7	100.0
Total	83	100.0	

Out of the testing techniques belonging to the black box method, the highest frequency of use is present in the Boundary Value Analysis (Tab. 4), while the lowest is in the Equivalence Class Partitioning (Tab. 5).

Out of the testing techniques belonging to the white box method, the highest frequency of use is present in the Condition Coverage (Tab. 6), and the lowest in the Code Complexity Analysis (Tab. 7).

Table 6 Distribution of use frequency of Condition Coverage

	Frequency	Percent	Cumulative percent
No	32	38.6	38.6
Yes	51	61.4	100.0
Total	83	100.0	

Table 7 Distribution of use frequency of Code Complexity Analysis

	Frequency	Percent	Cumulative percent
No	53	63.9	63.9
Yes	30	36.1	100.0
Total	83	100.0	

Data on the frequency of the use of individual software testing techniques in different sizes of surveyed organisations are shown in Fig. 7.

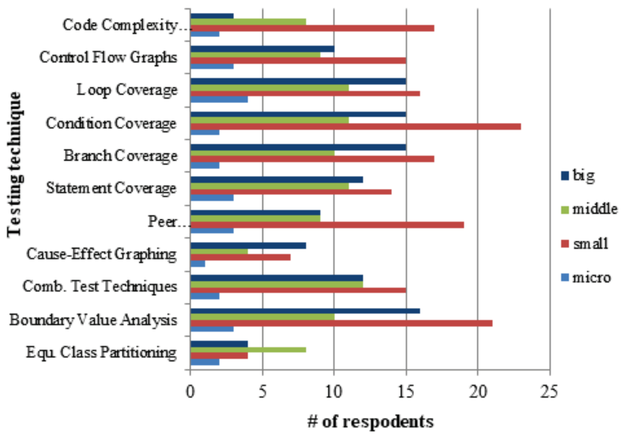

Figure 7 Frequency of the use of individual testing techniques in different organisation sizes

4.3.1 Similarities in Software Testing Techniques

The values of Jaccard index [32] between testing techniques are shown in Tab. 8. The values of this coefficient point to similarity between different software testing techniques. The highest index value (0.750) is present in Statement Coverage and Branch Coverage techniques.

Table 8 Testing technique similarity matrix (Jaccard index)

	ECP	BVA	CTT	CEG	PR	SC	BC	CC	LC	CFG	CCA
ECP	1.000	.333	.372	.226	.234	.261	.292	.232	.255	.250	.200
BVA	.333	1.000	.542	.321	.429	.500	.541	.603	.548	.450	.311
CTT	.372	.542	1.000	.452	.373	.500	.574	.484	.426	.418	.268
CEG	.226	.321	.452	1.000	.304	.333	.333	.268	.294	.357	.250
PR	.234	.429	.373	.304	1.000	.311	.355	.338	.365	.540	.273
SC	.261	.500	.500	.333	.311	1.000	.750	.685	.654	.351	.273
BC	.292	.541	.574	.333	.355	.750	1.000	.696	.636	.421	.276
CC	.232	.603	.484	.268	.338	.685	.696	1.000	.644	.354	.266
LC	.255	.548	.426	.294	.365	.654	.636	.644	1.000	.361	.288
CFG	.250	.450	.418	.357	.540	.351	.421	.354	.361	1.000	.340
CAA	.200	.311	.268	.250	.273	.273	.276	.266	.288	.340	1.000

Also, a high index value (0.696) is present in Branch Coverage and Condition Coverage techniques. Statement Coverage and Condition Coverage are techniques for which the index value amounts to 0.685. The Loop Coverage technique also has index value more than 0.65 in relation to the Statement Coverage, Branch Coverage and Condition Coverage techniques.

On the other hand, when it comes to the black box testing method, the only noteworthy similarity between testing techniques was identified between the Boundary Value Analysis and Combinatorial Test techniques with the index value of 0.542. However, what must be emphasized is the low index value of 0.333 between Equivalence Class Partitioning and Boundary Value Analysis techniques, which points to insufficient similarity of these techniques in surveyed organisations, although they are applied together almost as a rule at the theoretical level.

4.3.2 Results of the Cluster Analysis of Software Testing Techniques

How many homogeneous subsets are present in a set of software testing techniques was established by the use of hierarchical cluster analysis [33] (method: Average linkage, Euclidean distances for binary variables).

At the lowest linkage level, the cluster of Statement Coverage and Branch Coverage was formed. At subsequent linkage levels, they are extended by Condition Coverage, Loop Coverage, and Boundary Value Analysis, so that the first cluster at the highest level comprises a total of five testing techniques.

The second cluster at the lowest linkage level is formed by two pairs of techniques: Peer Reviews and Control Flow Graphs as one and Combinatorial Test and Cause-Effect Graphing as the other. At the subsequent level, it is extended by the Equivalence Class Partitioning, and the cluster is finally formed at the highest linkage level by adding the Code Complexity Analysis.

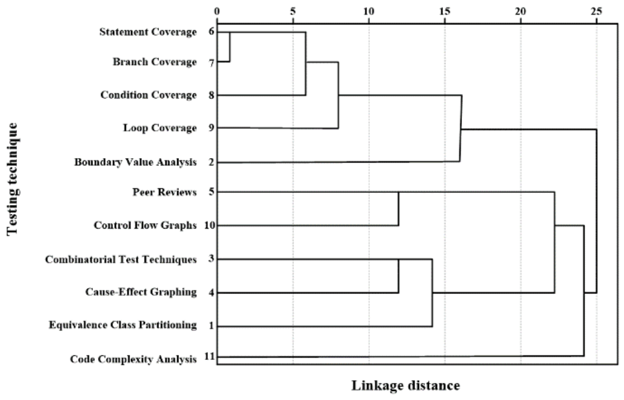

Figure 8 Software testing techniques dendrogram

It is interesting that the first cluster is possessed of high homogeneity in terms of belonging to the testing methods. Actually, four out of five testing techniques belong to the white box method (SC, BC, CC and LC), whereas the fifth technique, despite initially belonging to the black box testing method, can also be effectively used when testing software by white box testing method. Thus, a set of testing techniques is formed which are complementary, and significantly different from the testing techniques from the second cluster. The second cluster is possessed of a certain degree of heterogeneity in terms of type of testing methods to which the techniques belong (Fig. 8).

4.3.3 Results of MDS Software Testing Techniques

The multidimensional scaling method [34] was applied in order to present the similarities and differences between software testing techniques graphically, that is, to see what constellation is formed by testing techniques. The diagram shown in Fig. 9 shows the obtained layout of testing techniques in a two-dimensional space after the applied method. Closer points in the two-dimensional space testify to similarity between individual testing techniques, so that conditional groups of testing techniques can be formed, which are mutually similar.

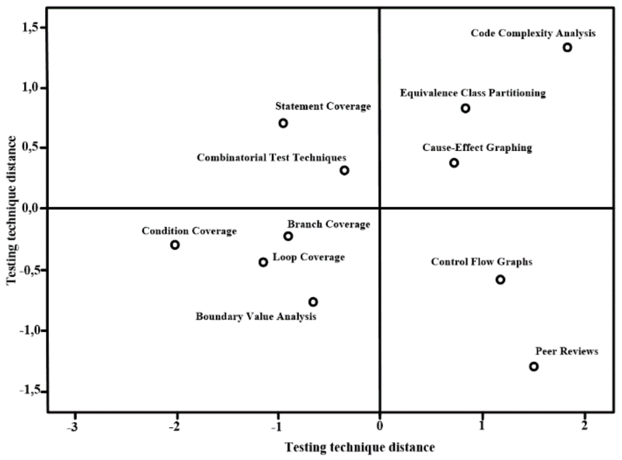

Figure 9 Configuration of testing techniques

The result of the multidimensional scaling method in Fig. 9 has confirmed the existence of the previously formed cluster comprising five software testing techniques: Condition Coverage, Loop Coverage, Branch Coverage, Statement Coverage and Boundary Value Analysis. Combinatorial testing represents a technique that could, after the application of this method, be appended to the previously formed cluster.

4.3.4 Probabilities of Application of Software Testing Techniques

A binomial test [35] has been applied in order to find out whether probabilities that an individual technique will be applied or not are equal or different. For Equivalence Class Partitioning, Cause-Effect Graphing, Condition Coverage and Code Complexity, whose value of asymptotic significance is equal to or less than 0.05, (0.000, 0.000, 0.048 and 0.015 respectively) it can be concluded that probability that an individual testing technique will be applied (1) or will not be applied (0) is not the same. On the other hand, for Boundary Value Analysis, Combinatorial Test Techniques, Peer Reviews, Statement Coverage, Branch Coverage, Loop Coverage and Control Flow Graphs, where the asymptotic significance value is higher than 0.05 (0.078, 1.000, 0.826, 0.826, 0.661, 0.380 and 0.380 respectively), it can be concluded that an individual testing technique will be applied (1) or will not be applied (0) is the same.

Cochran's *Q* test [36], whose results are shown in Tab. 9, was applied in order to find out whether the probabilities of application of eleven testing techniques (from Tab. 3) are equal or different.

Table 9 Cochran's *Q* test for eleven testing techniques

N	83
Cochran's *Q*	74.98882
df	10
p <	.000000

Null hypothesis "The probabilities of application of eleven software testing techniques are equal" was verified. Given that the value $Q = 74.98882 > X2$ ($df = 10$, $p < .000000$), the null hypothesis was rejected, which points to the conclusion that the probabilities of application of the eleven listed software testing techniques are not equal.

Cochran's *Q* test, whose results are shown in Tab. 10, was also applied in order to find out whether the probabilities of application of five most frequently used software testing techniques (Boundary Value Analysis, Combinatorial Test Techniques, Statement Coverage, Branch Coverage and Loop Coverage) are equal or different.

Table 10 Cochran's *Q* test for five most frequently applied testing techniques

N	83
Cochran's *Q*	5.406250
df	4
p <	.248097

Null hypothesis "The probabilities of application of five most frequently applied software testing techniques are not different" was verified. Given that the value $Q = 5.406250 < X2$ ($df = 4$, $p < .248097$), the null hypothesis was accepted, which points to the conclusion that the probabilities of application of the five most frequently applied software testing techniques are not different, that is, that the expectation that any of the five most frequent techniques will be applied is equal.

4.3.5 Exploratory Testing

To which extent respondents use the exploratory testing approach when conducting the testing process in organisations was checked by question 7 of the questionnaire. The scale frequencies are shown in Fig. 10.

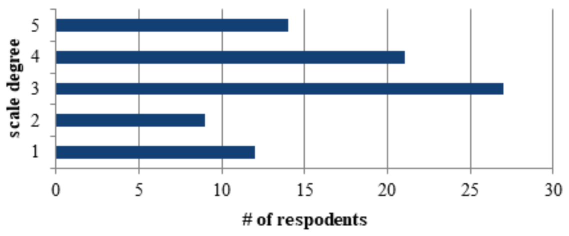

Figure 10 Scale frequencies for question 7

The largest number of respondents, 32.5%, expressed medium agreement with the statement that they use the exploratory testing approach when testing software products in organisations, whereas high and full agreement was expressed by 25.3% and 16.9% respondents respectively.

4.4 Formalization of Testing Methods and Techniques in Testing Models (Standards)

Whether the software testing methods and techniques are defined by testing model, i.e. standard used by software organisations, was established by question 8. A lower percentage of respondents (48.2%) in organisations do not have, while a slightly higher percentage of them (51.8%) have defined testing methods and techniques in the testing model or standard.

4.5 Results of the Qualitative Research Component

The qualitative research component provided input for thematic analysis of the content of respondents' replies for previously identified seven most frequently used software testing techniques identified in the realized survey. The results for each technique are shown in the following text.

Boundary Value Analysis generates a smaller number of test cases, enables reduction in software testing time, increases its efficiency and directly impacts the savings of financial resources in the testing process. It provides clear guidelines for generating test cases detecting potential defects in software, related to the defined limitations when data are entered by users. Also, it is necessary in order to shorten the time when testing equivalent classes. On the other hand, the defects do not necessarily have to be in the boundary values of variables, so that does not detect errors in algorithm logic. Furthermore, it is not suitable for testing different combinations of mutually dependent variables. In certain cases, the boundary value is hard to identify.

Combinatorial Test Techniques are mandatory due to tight deadlines and feature as the most used technique in medium-sized software organisations. Furthermore, it is used very successfully in non-functional platform testing for generating platform combinations (operative system and device) on which the software product will be tested, providing coverage of the target market. However, in complex software solutions the number of input combinations is fairly large even after its use, so that its application declines proportionally with the growth in software complexity. An exception is situations where it is profitable to initiate the automatization of test cases generated by this technique.

Peer Reviews provides higher reliability in the correctness and coverage of testing. It is the most effective in complex systems and systems where documentation is scarce. Owing to the inclusion of different perspectives, incomplete and unclear specification is quickly noticed and potentially immediately corrected. If it is realized informally, it can be carried out without much preparation. On the other hand, over formalization can be burdening and make the endeavour less productive. A very frequent occurrence when applying this technique is conflicts between team members due to different opinions.

Condition coverage enables detecting defects related to specific conditions in the program code, Statement Coverage ensures coverage of all statements in the program code, Loop Coverage enables identification of redundant loops, whereas Branch Coverage is most used for securing the execution of happy flow program scenario. They are simple both for adoption and for application. On the other hand, to be used efficiently, it is necessary to understand the program code, which is why it is most desirable for their implementation to be performed by developers. Also, it is not possible to ensure coverage of the program code and correctness of system functioning by using a single white box testing technique, because many defects then remain undetected.

5 DISCUSSIONS

Results of both qualitative and quantitative parts of the research are discussed in this section. Section 5.1 encompasses discussion on the results related to research questions RQ1, RQ2, and RQ4, i.e. the most frequently used software testing methods and techniques. Section 5.2 is a discussion on the results related to RQ3, formalization of testing techniques within an organisation, while section 5.3 points to the limitations of the empirical research.

5.1 The Most and Least Used Software Testing Methods and Techniques

Based on the data on frequency of use of software testing methods (Section 4.2.), a conclusion can be drawn that black box is slightly more dominantly used than white box testing method in software organisations in the Province of Vojvodina, which is an answer to RQ1. This result is similar to the results of similar surveys conducted in Australia and Canada [22, 25, 37]. A positive result of the survey is the information that the largest number of respondents in the Province of Vojvodina use both testing methods both in micro and in small, medium-sized and large software organisations, which increases the probability of identifying defects in software products.

As regards testing techniques (Section 4.3), two techniques have singled themselves out – Boundary Value Analysis and Condition Coverage, which are, each individually, used by more than 60% respondents in surveyed software organisations. Boundary Value Analysis is individually most used in large, whereas Condition Coverage is used in small software organisations. Such a result is not surprising, as these are "old", proven and widely known testing techniques. Actually, according to Myers [38], a large number of problems appear on the boundaries of linear variables, which was also the motive for the emergence of these techniques. According to the results of the latest survey [25], Boundary Value Analysis is also the most used testing technique in Canada whereas in Turkey this testing technique is second by use rate [26].

Combinatorial Test Techniques, Peer Reviews, Statement Coverage, Branch Coverage and Loop Coverage are techniques used, each individually by about 50% respondents in organisations. Together with the previously mentioned two, a set of seven most used software product testing techniques was formed which is an answer to RQ2.

Combinatorial Test Techniques provide a good cost/benefit ratio, because it requires a relatively small number of test cases (in relation to the total number) to provide a good coverage of the program code and identification of software defects, which is also stated in the results of studies appearing in publications [39, 40, 41].

In addition to being used as the golden standard for reviewing the program code and enabling improvement of its quality, peer reviews are also very suitable for reviewing the specification of the software product [42].

White box software testing techniques (SC, BC, CC and LC) are mutually complementary, which was confirmed by the result of hierarchical cluster and MDS analysis of quantitative data (Section 4.3.3), and they are therefore mostly used together in creating unit tests by developers.

After the application of Cochran's Q test to five most used testing techniques - BVA, CTT, BC, CC and LC (Section 4.3.4) - it was established that the probability of their use is not different, i.e., that there is equal expectation that any of these testing techniques will be applied.

However, sole application of one of the seven most used software-testing techniques identified by the research is very rare. Multiple techniques are most frequently used in parallel, as to ensure adequate code coverage (both in black box and white box testing techniques).

Since software development today predominantly follows agile methods, software testers decide on the best suited technique ad-hoc. This depends not only on technical features of the software, but also on indirect conditions. Often a technique best suited for a particular kind of software, cannot be applied for different factors, such as limited resources.

Although the remaining testing techniques (Equivalence Class Partitioning, Cause-Effect Graphing, Control Flow Graphs and Code Complexity Analysis) do not have the same frequency of use by the respondents as the seven most used testing techniques, it would be wrong to neglect them in the software testing process. The reason should be sought in the fact that certain testing techniques are efficient for the detection of certain software defects, but they are at the same time "blind" for detection of some other, more subtle software defects [43]. In comparison with Canada and Turkey [25, 26], the Equivalence Class Partitioning technique is less used, but it must be pointed out that this technique is used almost as a rule together with the complementary Boundary Value Analysis technique [38].

Testing techniques listed by respondents outside the offered ones are Ambiguity Review and Mutation Testing. Ambiguity Review is a technique used for testing requirements. Bearing in mind the results of study of Martin [44] that the largest number of software defects stem from poorly defined requirements, and that the process of correcting software defects is cheaper in the early phases of software development [45], this testing technique is recommended for use in the software product testing process. Mutation Testing (or fault injection), on the other hand, is a white box testing technique which is really insufficiently used in the Province of Vojvodina, which is also characteristic of software organisations in Canada and Australia [22, 25].

A positive result of survey research component is significant representation of the exploratory testing approach in software organisations in the Province of Vojvodina (Section 4.3.5). Compared to software organisations in Canada [25], exploratory testing is used more in the Province of Vojvodina. All the findings on the use of individual software testing techniques presented in the previous text constitute the answer to RQ4.

5.2 Formalization of Software Testing Techniques in Testing Model or Standard Used by the Organisation

The results of studies from 2004 and 2009 show that software organisations in Canada mostly rely on software testers' skills and intuition when conducting the testing process [23, 24]. One of the ways to avoid such a situation is to formalize testing techniques in testing models or standards in organisations, the way it was done in ISO/IEC/IEEE 29119 and TMap Next, which have these testing techniques and directions for use in their segments [18, 19]. Formalization should provide prerequisites for the application of all appropriate software testing techniques in conducting the testing process. Bearing this in mind, we wanted to find out what percentage of respondents in the Province of Vojvodina have formalized testing techniques in testing models or standards. The survey results are positive, as more than half of the respondents have formalized testing techniques which is an answer to RQ3, which creates prerequisites for the application of a larger number of appropriate and contemporary testing techniques when conducting the testing design sub-process. It is, however, very important to emphasize that the choice and application of testing techniques in the software testing process depends to a great extent on multiple factors, such as the nature of the software product, deadlines, human resources and the demanded test coverage.

5.3 Threats to Validity

The survey research strategy was conducted on a purposive sample of organisations in the Province of Vojvodina, accepting the risks entailed in this type of sampling. Also, the number of respondents in the sample of respondents was smaller by one than the a priori calculated number. In order to mitigate the risk pertaining to the organisation size, we included software organisations of different sizes into the sample. The sample of organisations included micro, small, medium sized and large software organisations. To ensure quality of data, we insisted on respondents who are currently or have at some time been employed on software product testing activities in the organisation. The fact that most persons involved in testing activities do not have a formal testing role (tester, test manager, test developer etc.) may, but does not have to be a limiting factor for data quality.

6 CONCLUSIONS AND FURTHER WORK

The survey results that we reached lead to a conclusion that respondents in software organisations in the Province of Vojvodina use to a significant extent most of the techniques belonging to ISO/IEC/IEEE 29119 testing standard when conducting the software testing process. Up to an extent, this can provide a satisfactory quality of realization of the testing design sub-process. The identified group of most used testing techniques (BVA, CC, LC, CTT, BC, PR and SC) together with the discussion on their specific features of application in the testing process, represents a practical recommendation for effective realization of the testing design sub-process. It is, however, important to emphasize that testing techniques that are less used must not be omitted from the testing design sub-process. It is therefore important to provide the formalization of all software testing techniques that need to be used when testing software products in organisations. Exploratory testing as an alternative approach to formal testing has a significant representation in surveyed software organisations.

Future research should move towards extending the area of surveyed organisations on the whole territory of the Republic of Serbia, as well as other countries in the region, so that the results of this research can be compared with other regions in the country, and later on, the results of the Republic of Serbia with the results of the countries of the region. We are also planning to extend the research subject, which will cover the complete testing process, so that we can compare these results with the results of similar studies.

Acknowledgements

We are indebted to the ICT Cluster Vojvodina for their support in establishing contact with the organisations included in the survey.

7 REFERENCES

[1] Wieczorek, M., Vos, D., & Bons, H. (2014). *Systems and Software Quality, The next step for industrialization*. Berlin Heidelberg: Springer-Verlag. https://doi.org/10.1007/978-3-642-39971-8
[2] Origo (2014). Szoftverhiba miatt 1 pennyért adta termékeit az Amazon. Retrieved from http://www.origo.hu/techbazis/20141215-szoftverhiba-miatt-lehetett-1-pennyert-vasarolni-az-amazonnal.html
[3] Focus (2012). Alitalia verschenkt aus Versehen Tausende Gratis-Tickets. Retrieved from http://www.focus.de/reisen/flug/wegen-softwarefehler-alitalia-verschenkt-aus-versehen-tausende-gratis-tickets_aid_844004.html
[4] Heusser, M. (2012). Software Testing Lessons Learned From Knight Capital Fiasco. Retrieved from http://www.cio.com/article/713628/Software_Testing_Lessons_Learned_From_Knight_Capital_Fiasco
[5] Zeti (2015). Sparkassen-Automaten funktionieren wieder. Retrieved from http://www.zeit.de/wirtschaft/2015-09/sparkasse-geldautomaten-ausfall
[6] News (2011). Two men arrested over Commonwealth Bank's ATM free-for-all. Retrieved from http://www.news.com.au/finance/money/computer-glitch-hits-cba-customers/story-e6frfmcr-1226014261756
[7] BBC (2015). Divorce form error 'could have led to unfair settlements'. Retrieved from http://www.bbc.com/news/uk-35128010
[8] Fung, B. (2014). How a dumb software glitch kept thousands from reaching 911. Retrieved from https://www.washingtonpost.com/news/the-switch/wp/2014/10/20/how-a-dumb-software-glitch-kept-6600-calls-from-getting-to-911/
[9] Tsvirko, N. (2015). Department of Education blames an IT glitch for over 600 child abuse reports not reaching Queensland police. Retrieved from http://www.dailymail.co.uk/news/article-3180864/Department-Education-blames-glitch-600-child-abuse-reports-not-reaching-Queensland-police.html
[10] Index (2015). Szoftverhiba miatt korábban engedtek ki 3200 elítéltet. Retrieved from http://index.hu/tech/2015/12/23/szoftverhiba_miatt_korabban_engedtek_ki_3200_eliteltet/
[11] Williams, R. (2015). Apple's watchOS2 software delayed due to bug. Retrieved from http://www.telegraph.co.uk/technology/apple/watch/11869281/Apples-watchOS-2-software-delayed-due-to-bug.html
[12] Stern (2014). Audi ruft 850.000 A4 wegen Softwarefehler zurück. Retrieved from http://www.stern.de/auto/news/airbag-probleme--audi-ruft-850-000-a4-in-die-werkstaetten-zurueck-3843722.html

[13] Guardian (2016). Error 53' fury mounts as Apple software update threatens to kill your iPhone 6. Retrieved from http://www.theguardian.com/money/2016/feb/05/error-53-apple-iphone-software-update-handset-worthless-third-party-repair

[14] Baraniuk, C. (2016). Twitter network down for many users after technical fault. Retrieved from http://www.bbc.com/news/technology-35351154

[15] Britton, T., Jeng, L., Graham, C., Cheak, P., & Katyenellenbogen, T. (2015). Reversible Debugging Software, University of Cambridge. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.370.9611&rep=rep1&type=pdf

[16] Tricentis (2016). Software Fail Watch: 2015 in Review. Retrieved from http://d1n3e8fpzha6ub.cloudfront.net/wp-content/uploads/2016/01/19135532/Tricentis_Software_Fail_Watch_20151.pdf

[17] Afzal, W. & Torkar, R. (2008).Incorporating Metrics in an Organisational Test Strategy. *Software Testing Verification and Validation Workshop, ICSTW '08* (pp. 304-315). Washington: IEEE Computer Society. https://doi.org/10.1109/ICSTW.2008.23

[18] ISO/IEC/IEEE 29119:2013. Software Testing. Retrieved from http://www.softwaretestingstandard.org

[19] Koomen, T., van der Aalst, L., Broekman, B., & Vroon, M. (2006). *TMap Next*.Hertogenbosch: UTN Publishers.

[20] TMMi Foundation (2010). *Test Maturity Model integration (TMMi) v3.1.* Dublin: TMMi Foundation.

[21] Geetha Devasena, M. S., Gopu, G., & Valarmathi, M. L. (2016). *International Journal of Software Engineering and Knowledge Engineering*, *26*(1), 1-13. https://doi.org/10.1142/S0218194016500017

[22] Ng S. P., Murnane, T., Reed, K., Grant, D., & Chen, T. Y. (2004). A Preliminary Survey on Software Testing Practices in Australia. *Proceedings of the 2004 Australian Software Engineering Conference (ASWEC'04).* London: IEEE Computer Society. https://doi.org/10.1109/ASWEC.2004.1290464

[23] Geras, A. M., Smith, M. R., & Miller, J. (2004). A survey of software testing practices in Alberta. *Canadian Journal of Electrical and Computer Engineering, 29*(3), 183-191. https://doi.org/10.1109/CJECE.2004.1532522

[24] Garousi, V. & Varma, T. (2010). A replicated survey of software testing practices in the Canadian Province of Alberta: what has changed from 2004 to 2009? *Journal of Systems and Software, 83*(11), 2251-2262. https://doi.org/10.1016/j.jss.2010.07.012

[25] Garousi, V. & Zhi, J. (2013). A survey of software testing practices in Canada. *Journal of Systems and Software, 83*(11), 1354-1376. https://doi.org/10.1016/j.jss.2012.12.051

[26] Gaorousi, V., Coskuncay, A., Betin-Can, A., & Demirors, O. (2014). A Survey of Software Engineering Practices in Turkey. Retrieved from http://arxiv.org/ftp/arxiv/papers/1412/1412.4648.pdf

[27] ISO/IEC/IEEE 29119-4:2013. Test Techniques. Retrieved from http://www.softwaretestingstandard.org/part4.php

[28] Garcia, C., Dávila, A., & Pessoa, M, (2014). Test Process Models: Systematic Literature Review. *14th International Conference, SPICE 2014,* (pp. 84-93). Berlin Heidelberg: Springer International Publishing. https://doi.org/10.1007/978-3-319-13036-1_8

[29] Cohen, J. A. (1992).Power Primer. *Psychological Bulletin, 112*(1), 55-159. https://doi.org/10.1037/0033-2909.112.1.155

[30] Dattalo, P. (2008). *Determining Sample Size: Balancing Power, Precision, and Practicality*. Oxford: Oxford University Press, Inc. https://doi.org/10.1093/acprof:oso/9780195315493.001.0001

[31] European Commission (2003). The New SME Definition - User guide and model declaration. Retrieved from http://www.eusmecentre.org.cn/sites/default/files/files/news/SME%20Definition.pdf

[32] Gordon, A. D. (1981). *Classification: Methods for the exploratory analysis of multivariate data*. London: Chapman & Hall/CRC Monographs on Statistics & Applied Probability.

[33] Everitt, B. (2011). *Cluster Analysis* (5th ed.).Chichester: John Wiley & Sons, Ltd. https://doi.org/10.1002/9780470977811

[34] Borg, I. & Groenen, P. J. F. (2005). *Modern Multidimensional Scaling: Theory and Applications* (2nd ed.). New York: Springer Science + Business Media, Inc.

[35] Gibbons, J. D. (1976). *Nonparametric Methods for Quantitative Analysis.* New York: Holt, Rinehart and Winston.

[36] Israel, D. (2008). *Data Analysis in Business Research: A Step-by-Step Nonparametric Approach*. Thousand Oaks, Ca: SAGE Publications Inc.

[37] Taipale, O., Smolander, K., & Kälviäinen, H. (2005). Finding and ranking research directions for software testing. *European Conference on Software Process Improvement* (pp. 39-48). Berlin Heidelberg: Springer-Verlag. https://doi.org/10.1007/11586012_5

[38] Myers, G. (2004). *The Art of Software Testing* (2nd ed.). New Jersey: John Wiley & Sons.

[39] Grindal, M., Lindström, B., Offutt, J., & Andler, S. F. (2006). An evaluation of combination strategies for test case selection. *Empirical Software Engineering, 11*(4), 583-611. https://doi.org/10.1007/s10664-006-9024-2

[40] Copeland, L. (2004). *A Practitioner's Guide to Software Test Design*. Boston, London: Artech House Publishers.

[41] Page, A., Johnston, K., & Rollison, B. (2009). *How We Test Software at Microsoft*. Washington: Microsoft Press.

[42] Graham, D. (2002). Requirements and Testing: Seven Missing-Link Myths. *IEEE Software, 19*(5), 15-17. https://doi.org/10.1109/MS.2002.1032845

[43] Beizer, B. (1995). *Black-Box Testing: Techniques for Functional Testing of Software and Systems*. New York: John Wiley. https://doi.org/10.1109/MS.1996.536464

[44] Martin, J. (1984). *An Information Systems Manifesto*. New Jersey: Prentice-Hall.

[45] Boehm, B. & Basili, V. (2001). Software Defect Reduction Top 10 List. *IEEE Computer, 34*(1), 135-137. https://doi.org/10.1109/2.962984

Contact information:

Vuk VUKOVIC, PhD, Assistant Professor
University of Novi Sad, Faculty of Economics in Subotica,
Segedinski put 9-11, 24000 Subotica, Serbia
E-mail: vuk.vukovic@ef.uns.ac.rs

Jovica DJURKOVIC, PhD, Full Professor
University of Novi Sad, Faculty of Economics in Subotica,
Segedinski put 9-11, 24000 Subotica, Serbia
E-mail: djovica@ef.uns.ac.rs

Marton SAKAL, PhD, Full Professor
University of Novi Sad, Faculty of Economics in Subotica,
Segedinski put 9-11, 24000 Subotica, Serbia
E-mail: marton@ef.uns.ac.rs

Lazar RAKOVIC, PhD, Assistant Professor
(Corresponding author)
University of Novi Sad, Faculty of Economics in Subotica,
Segedinski put 9-11, 24000 Subotica, Serbia
E-mail: lazar.rakovic@ef.uns.ac.rs