JOURNAL OF SOFTWARE ENGINEERING & INTELLIGENT SYSTEMS
ISSN 2518-8739
31 August 2020, Volume 5, Issue 2, JSEIS, CAOMEI Copyright © 2016-2020
www.jseis.org

WEIGHTED NETWORKS IN SOFTWARE MAINTENANCE INDUSTRIAL PRACTICE: A PRELIMINARY LITERATURE REVIEW

ZELJKO STOJANOV [1], JELENA STOJANOV [2]

[1,2] University of Novi Sad, Technical Faculty "Mihajlo Pupin", Zrenjanin, Serbia
Email: zeljko.stojanov@uns.ac.rs [1], jelena.stojanov@uns.ac.rs [2]

OPEN ACCESS

Academic Editor: Imran Ghani, USA.
Published: 31 August, 2020
Funding: Author has no funds.
Article Type: Review Article

ABSTRACT

Software maintenance industrial practice should ensure reliable software systems that operate in demanding business conditions. Efficient software maintenance requires methods and tools, efficient organization and skilled maintainers. Development and further improvement of maintenance methods and tools require comprehensive analysis of software systems, as well as human and organizational issues. Weighted networks are efficient tool that enable analysis of different aspects of software maintenance, both technical and non-technical. This article presents a preliminary literature review of weighted networks applications in software maintenance industrial practice. Literature review process, findings based on the proposed research questions, and discussion of challenges, validity and research implications are presented.

Keywords: software maintenance; industrial practice; weighted networks; weighted graphs; literature review;

1. INTRODUCTION

Contemporary business is highly dependent on software systems, which puts great demand for efficient software maintenance. However, many software organizations do not have defined maintenance processes, which causes problems for organizations that need reliable and efficient software systems [1]. Maintenance of software systems is not easy because it requires efficient management in software organizations that optimize costs and minimize risks of maintenance activities. Software maintenance is as a set of complex and demanding activities after software product delivery aimed at modifying software product in order to improve performance, correct faults or adapt to modified environment [2].

Software systems evolve during the operation phase, which causes several unpredictable changes making software systems difficult to understand and maintain [3], requiring skilled and knowledgeable maintainers [4]. Since software system changes during its life cycle it is important to perform adequate maintenance activities on software components affected by the change. According to Yu et al. [5] it is important to identify elements of software that freequently change because in object-oriented software systems about the half of classes never change. In this way, cost and risks of maintenance activities can be minimized. This asumes accurate and reliable models of software systems and well organized maintainers in software organizations.

Networks or graphs [6] enable representation of software systems in a natural way, where software elements are presented as nodes and their relations are presented as edges (links). Since a representation of a software system structure with a network is easy to understand and comprehend, techniques from network theory can be easily implemented for analyzing properties of software systems. Networks can be also used as an efficient tool for analyzing software maintenance and evolution processes, with the focus on stability of software structure that evolves according to the changed needs of software users [7].

Systematic literature reviews have become a common way for analyzing and integrating evidence-based knowledge on software engineering practice [8][9]. Literature reviews have been conducted on different topics in software maintenance, such as management of global software maintenance in distributed settings [10], understanding change attributes and measures in software maintenance and evolution [11], code-based change impact analysis techniques [12], software quality attribute of maintainability in process of software maintenance [13], or software fault detection and correction process, models and techniques [14]. Li et al. [15] presented a review of inter-disciplinary research on using complex networks in software engineering for period from 2013 to 2019. The focus was on modeling static and dynamic properties of software structure, as well as on application of complex networks in real software systems for defining metrics, predicting faults, refactoring, and identifying

Corresponding Author: Zeljko Stojanov
Email Address: zeljko.stojanov@uns.ac.rs

elements of software structure. However, there is a lack of literature reviews focused on using weighted networks in software maintenance.

Based on the presented discussion, this article aims at presenting a preliminary review of empirical studies on using weighted networks in software maintenance industrial practice. The rest of the article is structured as follows. The second section presents a short overview of weighted networks. The third section outlines a literature review study, while the fourth section presents discussion of identified challenges, limitations and validity of the literature review study, and implications for practitioners from industry, researchers and educators. The last section contains concluding remarks and future research directions.

2. WEIGHTED NETWORKS

Weighted graphs that model real phenomena are commonly named weighted networks. Weighted graph contains vertices (nodes), edges (links) and a function that quantifies a feature of edges. Abstract definition of the notion is based on the set theory.

Weighted network is the triple G=(V,E,*w*), whose elements satisfy the following properties:

- $V=\{v_i \mid i \in I\}$ is the set of nodes (vertices). Amount of nodes |V| is order of the network. Nodes in the network represent real objects (devices, agents, persons, elements of an algorithm, events).
- $E=\{e_a \mid a \in A\}$ is the set of edges. Network size is amount of edges |E|. Network edges present a relation among nodes (connections, collaborations, links, order). Precisely, each edge is a pair of nodes $e_a=(v_i,v_j)$. The edge e_a is incident with the nodes v_i and v_j. The node v_i is adjacent to v_j. If the relation presented by the edges is symmetric, the nodes v_i and v_j are mutually adjacent.
- $w:E\rightarrow[0,\infty)$ is the weight function that assigns to each edge a number quantifying its weight, $w_{ij} = w(e_a) = w(v_i,v_j)$. The new label contains information of the nodes connected by the edge, so it is commonly used. In the case that all edges have the same weight, the weight function is trivial, hence can be omitted and the weighted graph becomes the graph G=(V,E).

It could be of interest to consider only a part of the weighted network, some of the vertices $V_1 \subset V$ and some of the edges, with overtaken their weights, incident with elements of V_1 contained in $E_1 \subset E$. Obtained network $G_1=(V_1,E_1,w_{E1})$ is named as a subgraph or subnetwork of the starting one G.

Basic topological properties of weighted network will be presented, due to their importance for the topic of this article.

The number of edges which are incident with a node v_i is its degree $d(v_i)$. A set of nodes adjacent to the node v_i is the set of neighbours $N(v_i)$, and it contains exactly $d(v_i)$ elements.

A path in the network is a sequence $v_0, e_1, v_1, e_2, \ldots, v_k$, where each edge e_a, $a = 1, \ldots, k$ is incident with the nodes v_{a-1} and v_a. The starting node is v_0, and the target node is v_k. If the relation among nodes is symmetric the path is reversible. As the path is fully determined with the listed nodes, the edges in the sequence are commonly omitted. The length of the path is the number of edges it contains. Exclusively seen, each path is a trivial subnetwork.

Two nodes v_i and v_j are connected if there is a path with the starting node v_i and the target node v_j. The length of the shortest path between the nodes v_i and v_j is their distance $d(v_i, v_j)$. In the weighted network, a path between any two vertices v_i and v_j has, besides the length, its weight, obtained as a sum of the weights of all edges in the path. The optimal (shortest) path in weighted network implies the minimum weight.

The network is connected if each two nodes are connected. Stronger connectivity accelerates all procedures within the network and enhances flow (of information).

Importance of a node in connected network reflects its significance in the connectivity, and is presented by the notion of centrality. There are more kinds of the centrality dependent upon the network context. All of them provide analogous information: if a node has big centrality measure, its removal from the network significantly endangers the connectivity and reduces the flow.

More details on weighted networks and their applications within software engineering can be found in [16].

3. PRELIMINARY LITERATURE REVIEW

The literature review study presents a preliminary review of empirical studies that deal with application of weighted networks (weighted graphs) in software maintenance industrial practice. The method for the preliminary

literature review is a modification and simplification of a systematic literature review protocol that is usually used for comprehensive literature review of specific issues in software engineering [17]. The modified method, presented in Figure 1, contains the following steps: determining literature review objective, setting criteria for the selection of studies, searching for studies, identifying relevant studies, detailed analysis of the selected studies, and synthesizing study findings.

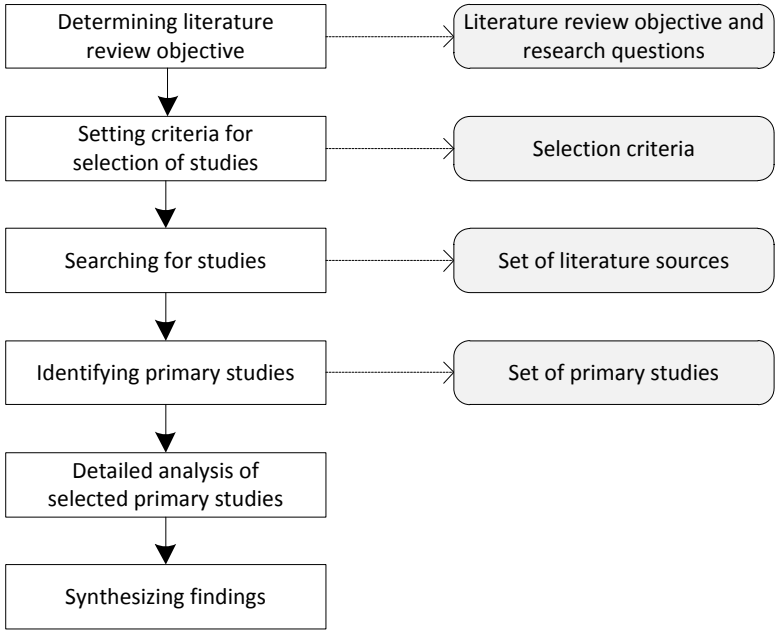

Figure 1. Overview of a method for preliminary literature review

Based on the detailed review of the selected studies and synthesized findings, challenges and implications are identified.

3.1 Study objectives

The main objective of the study is to identify problems and challenges based on empirical evidence from software maintenance practice in industry. This ensures that only studies that report evidence from the industry can be included in further analysis. The following research questions (RQ) are proposed:

RQ1: Which maintenance activities are addressed in the identified studies?

RQ2: Which software types are targeted in the identified studies?

RQ3: Which abstraction level of software systems is used?

RQ4: How weighted networks are created?

RQ5: Which elements are used in created weighted networks?

3.2. Selection criteria for primary studies and search

Based on the proposed objectives, the following criteria for the selection of the relevant studies were proposed:

1. Study reports experience from industry
2. Study relates to software maintenance phase in software life cycle.
3. Weighted networks are used for modelling and analyzing segments of practice.
4. The study is written in English.
5. The study is published in conference proceeding or journal with the strictly defined review process.

The initial set of literature sources was obtained by search using Google Scholar, a free web search engine that indexes the scholar literature. The search was constrained to the last ten years (2011 - 2020), which ensures that the findings present an up-to-date state of the industrial practice. The search was performed by using keywords "software maintenance" and "weighted network", combined in the search string by using logical AND operator, as follows:

"software maintenance" AND "weighted network"

An additional search, which includes analysis of references and citations of the best ranked literature sources, listed by relevance from Google Scholar (sort by relevance), was performed on the pages of publishers where literature sources are available, such as IEEE, ACM, Hindawi, World Scientific, and ScienceDirect. Detailed bibliographic data on the literature sources was obtained from the publisher's website.

3.3 Identification of primary studies

Through the search on Google Scholar, 41 literature sources were identified, from which 12 were selected as primary studies relevant for further analysis. Citations for primary studies (PS01 to PS12) are presented in Table 1.

Table 1. Primary studies identified through searching literature sources

No.	Citation of primary study
PS01	Chong, C. Y., & Lee, S. P. (2015). Analyzing maintainability and reliability of object-oriented software using weighted complex network. Journal of Systems and Software, 110, 28-53.
PS02	Ding, Y., Li, B., & He, P. (2016). An improved approach to identifying key classes in weighted software network. Mathematical Problems in Engineering, 2016.
PS03	Guiying, L., Fei, X., Binbin, L., Xiaolin, Z., & Cuicui, C. (2016). Research of software defect prediction based on complex network. In Electrical and Control Engineering & Materials Science and Manufacturing: The Proceedings of Joint Conferences of the 6th (ICECE2015) and the 4th (ICMSM2015) (pp. 332-342).
PS04	He, H., Shan, C., Tian, X., Wei, Y., & Huang, G. (2018). Analysis on influential functions in the weighted software network. Security and Communication Networks, 2018.
PS05	Pan, W., Li, B., Liu, J., Ma, Y., & Hu, B. (2018). Analyzing the structure of Java software systems by weighted K-core decomposition. Future Generation Computer Systems, 83, 431-444.
PS06	Wang, Q., Shan, C., Zhao, X., Dong, J., Ren, J., & Liu, J. (2019). A Novel Algorithm for Identifying Key Function Nodes in Software Network Based on Evidence Theory. International Journal of Software Engineering and Knowledge Engineering, 29(03), 415-432.
PS07	Chong, C. Y. (2016). Constrained clustering approach to aid in remodularisation of object-oriented software systems/Chong Chun Yong (Doctoral dissertation, University of Malaya).
PS08	Li, H., Hao, L. Y., & Chen, R. (2016). Multi-level formation of complex software systems. Entropy, 18(5), 178.
PS09	Teixeira, J., Leppänen, V., & Hyrynsalmi, S. (2020). Network Science, Homophily and Who Reviews Who in the Linux Kernel?. In Proceedings of the 28th European Conference on Information Systems (ECIS), An Online AIS Conference, June 15-17, 2020. https://aisel.aisnet.org/ecis2020_rp/104
PS10	Concas, G., Marchesi, M., Monni, C., Orrù, M., & Tonelli, R. (2017). Software quality and community structure in java software networks. International Journal of Software Engineering and Knowledge Engineering, 27(07), 1063-1096.
PS11	Badashian, A. S. (2016, May). Realistic bug triaging. In 2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C) (pp. 847-850). IEEE.
PS12	Chen, L., Qian, J., Zhou, Y., Wang, P., & Xu, B. (2014). Identifying extract class refactoring opportunities for internetware. Science China Information Sciences, 57(7), 1-18.

The reason for exclusion of 29 studies from further analysis is due to their non-fulfilment of the following criteria: 17 studies do not relate to software maintenance, 5 studies do not use weighted networks, 2 studies are written in Chinese, 2 studies are published as not reviewed, 2 studies present the same results as the primary study PS05 but are published earlier, and primary study PS07 is listed two times as published at two different Web addresses.

3.4 Detailed analysis of primary studies

Analysis of the searched studies was conducted in two steps: (1) preliminary review of title, abstract, keywords and conclusions, and (2) detailed analysis of the whole text of the study.

Through preliminary review 14 studies were selected as primary studies suitable for more detailed analysis. However, through detailed analysis of the whole text of the studies, two studies were excluded from the analysis because they are related to software development phase, while only potential benefits for maintenance activities were mentioned. Finally, 12 studies fulfil all requirements to be included in the detailed analysis and synthesis of general findings. Detailed analysis of the studies was based on the proposed research questions, and includes:

- Analysis of the methodology presented in the study, aimed at identifying the way for creating weighted network for the specific problem in software maintenance.
- Identification of specific maintenance activities addressed in the study.
- Determination of software types for which study is suitable.
- Determination of abstract level of software systems used in empirical studies.
- Identification of entities used in created weighted networks.

3.5 Synthesizing findings

The synthesized findings of this study are organized around proposed research questions (RQ). Referencing of the identified primary studies is presented with their numbers PS01 to PS12, while details on these studies can be accessed by using citations in Table 1.

RQ1: Detailed insight into the selected studies revealed that different maintenance activities are addressed. Distribution of identified primary studies according to addressed maintenance activities is presented in Table 2.

Table 2. Distribution of primary studies per identified maintenance activities

Maintenance activity	**Primary studies**
Estimation of maintenance effort	PS01
Identification of classes prone to bugs and errors	PS01, PS03
Identification of key classes for maintenance	PS02
Software update	PS04, PS06
Software understanding and comprehension	PS05, PS07, PS08
Fault detection in source code	PS06
Reengineering	PS07
Refactoring	PS08, PS12
Adaptation	PS08
People organization	PS09, PS11
Identification of modules prone to defects	PS10
Bug assignment to developers	PS11

The complexity of maintenance practice in software industry is reflected in variety of addressed maintenance activities related to both technical and human side of the practice. Software understanding and comprehension are identified as activities addressed in three studies, which is expected since comprehension and understanding are the most demanding activities with the highest costs [18].

RQ2: Detailed analysis points out that majority of the selected primary studies use open source software for empirical research because they are free and easily accessible. Data sets are collected from bug tracking systems, systems for versioning control, or public forums focused on software engineering community. In some studies, presented method and findings are based on analyzing many subsequent versions of the same software system, such as *Eclipse* and *NetBeans* in PS10, *JEdit* in PS03, *tar* and *cflow* in PS04, or Java complex software systems in PS05. In Table 3, distribution of primary studies per identified types of software systems used as objects in empirical studies is presented.

Table 3. Distribution of primary studies per identified types of used software systems

Type of software systems used in studies	**Primary studies**
Object oriented software	PS01, PS02, PS03, PS05, PS07, PS08, PS10, PS12
C/C++ programs	PS04, PS06
Linux kernel	PS09

RQ3: Constructed weighted networks for software systems present software at different levels of abstraction, depending on the proposed objectives in the analyzed studies. Table 4 presents software abstraction levels used in primary studies.

Table 4. Distribution of primary studies used abstraction level of software systems

Abstraction level of software systems used in studies	Primary studies
Level of classes in object-oriented design	PS01, PS02, PS03, PS05, PS07, PS08, PS10
Level of functions or methods in source code	PS04, PS06, PS12
Design pattern level	PS08
Software component level	PS08

Primary studies PS09 and PS11performed analysis on the networks focused on human issues in software maintenance, such as assignments of maintenance tracking of software subsystems to maintainers in PS09, and maintainers' expertise in PS11. Primary study PS08 used presentation of software systems at three levels (classes, design patterns, components) in order to get more realistic insight into comprehension and refactoring activities.

RQ4: Different approaches are used for creating weighted networks, which is presented in Table 5. Many approaches are based on several steps for converting source code or software models into weighted networks, while the authors used available software libraries or develop own software tools as an aid in this process (e.g. parsers, converters between different formats and representations).

Table 5. Approaches in creating weighted networks

Approach	Primary studies
Convert object-oriented code to class diagrams and further to weighted networks	PS01, PS07
Convert object-oriented code to XML file and further to weighted networks	PS02
Parsing source code	PS03, PS05, PS10, PS12
Analysis of multiple software execution traces	PS04, PS06
History data about software evolution	PS08
Commit messages on software maintenance	PS09
Stack Overflow tags on maintainers expertise	PS11

Approaches focused on software structure create weighted networks by analyzing and modelling software systems at different levels of granularity, depending on the proposed objectives, while approaches focused on human and organizational issues create networks by analyzing additional data related to maintenance activities available in issue tracking systems or software engineering forums.

RQ5: Depending on the proposed study objective and used approach, different elements are used as nodes and edges in created weighted networks, which are presented in Table 6.

Table 6. Elements used as nodes and edges in created weighted networks

Nodes and edges in created weighted networks	Primary studies
Classes and relations between them	PS01, PS02, PS03, PS05, PS07, PS10
Software functions and calls among them	PS04, PS06, PS12
Depends on the granularity level	PS08
Maintainers and their relations	PS09
Stack Overflow tags about maintainers expertise	PS11

4. DISCUSSION

In this section, identified challenges, an analysis of the validity of the proposed method of literature analysis and the findings, and research implications are discussed.

4.1 Identified challenges

The first challenge relates to the creation of a network for a particular software maintenance problem. The creation of a network as a mathematical model imposes exact and distinct determination of objects which would be represented as nodes and edges, together with identification of a method or a function that will assign weights to edges. When research is focused on open source software systems, whose history of development and maintenance is easily accessible, creation of weighted networks can be easily done by using available or by

developing own software tools. However, when the focus is on commercial and licensed software systems, data sets are much harder to access and usually require quite specific approaches and tools for creating networked representations of investigated phenomena.

The second challenge relates to the objects that are inquired by using weighted networks. All identified primary studies except PS09 and PS11 deal with technical issues and are focused on software structure, while studies PS09 and PS11 are focused on human and organizational issues. Since software engineering can be observed as socio-technical system composed of technical, organizational and human entities with complex relationships [19][20], it is necessary to adopt multidisciplinary approach to analysis of software maintenance practice and create networks that represent problems in maintenance practice with heterogeneous networks [21]. This will require composing research teams with experts from different technical and social disciplines [22], which will enable deeper understanding of observed phenomena.

4.2 Limitations and validity

Two limitations of this study arise from its quite simplified design, which impacts validity of the study findings.

The first limitation relates to selecting the keywords for searching for literature sources and forming the string for searching literature sources. Only two keywords are used for creating one search string, although guidelines for literature search [8][17] suggest using multiple keywords and their synonyms, which enables creation of more complex and comprehensive search strings, resulting in a more reliable search of literature sources and more comprehensive findings.

The second limitation relates to the selection of literature sources to be searched. For this preliminary literature review, Google Scholar is selected, while common digital libraries with software engineering literature such as IEEE, ACM, Springer, Wiley, ScienceDirect are not searched. This limitation leads to identification of only 12 primary studies. In addition, there is an obvious lack of studies published in leading software engineering journals with high impact factor, published by leading publishers. This limitation will be addressed in future work through organization of a systematic literature review study by following proposed guidelines.

The validity of the presented study can be judged in terms of internal and external validity, which are commonly used in empirical software engineering [23]. Internal validity, related to soundness of research process. It is addressed through detailed description of the study design and findings based on the proposed research questions. External validity relates to implementing research process or findings in other settings. This can be accomplished by following described research process, selecting appropriate keywords for search, and selecting inclusion/exclusion criteria for a specific research topic (segment of practice).

4.3 Research implications

Managers and software experts from industry can find a short summary of topics related to use of weighted networks as efficient tools for solving maintenance problems in practice. For each topic, deeper insight into the selected primary studies will help in identifying the most appropriate methods and tools for solving software maintenance problems by using weighted networks, or will provide ideas and directions how to tackle specific practical problems.

University educators can find systematization of topics in software maintenance practice around proposed research questions, and include them in curricula of software engineering related subjects [9]. This will enable introduction of attractive and up-to-date problems from real industrial practice to students, helping them to be more prepared for real industrial projects.

Researchers can find guidelines for organizing preliminary literature review on the selected topic from the industrial practice, or can use some findings as a starting point for further research projects. Presented study design and findings can be particularly useful for PhD students, who can find details on organizing preliminary literature reviews, which are helpful in the beginning phase of PhD research.

5. CONCLUSIONS

This article presents a preliminary literature review on using weighted networks in industrial software maintenance practice. The findings are organized around the proposed research questions, providing a short summary of identified topics in software maintenance practice.

Despite the stated limitations, this study has its own contributions. The first contribution is a detailed description of the a preliminary literature review process, which is commonly used as a preparation phase for conducting more comprehensive and systematic literature review in the selected field. The second contribution is a short summary of identified topics related to use of weighted networks in software maintenance, which can be

used as starting points for further research. The third contribution of this study is discussion of implications for practitioners from industry, researchers and educators.

Future research will be directed towards preparing and conducting a more detailed systematic literature review, which will be used for identifying current research trends, proposing attractive directions for further research, and proposing potential improvements of university teaching based on contemporary topics in software maintenance industrial practice.

ACKNOWLEDGEMENT

Ministry of Education, Science and Technological Development, Republic of Serbia, supports this research under the project "The development of software tools for business process analysis and improvement", project number TR32044.

REFERENCES

1. April, A., and A. Abran, *Software Maintenance Management Evaluation and Continuous Improvement*. 2008: IEEE Computer Society & Wiley. Hoboken, NJ, USA.
2. Bourque, P., and R.E. Fairley (ed.), *Guide to the Software Engineering Body of Knowledge (SWEBOK), 3rd ed.* 2014: IEEE Press. Piscataway, NJ, USA.
3. Chen, L., J. Qian, Y. Zhou, P. Wang, B. Xu, (2014). Identifying extract class refactoring opportunities for internetware. *Science China Information Sciences*, 2014. 57(7), 1-18.
4. Stojanov, Z. Thematic knowledge framework on human factor in software maintenance practice: A study in a micro software company. *Journal of Software Engineering & Intelligent Systems*, 2019. 4(1), p. 41-57.
5. Yu, L., Y. Li, and S. Ramaswamy, Using peer comparison approaches to measure software stability. *Journal of Software Engineering & Intelligent Systems*, 2018. 3(1), p. 45-53.
6. Gross, J. L., and J. Yellen (eds), *Handbook of graph theory*. 2003: CRC Press, Taylor & Francis Group. Boca Raton, FL, USA.
7. Jenkins, S., and S.R. Kirk, Software architecture graphs as complex networks: A novel partitioning scheme to measure stability and evolution. *Information Sciences*, 2007. 177(12), 2587-2601. doi: 10.1016/j.ins.2007.01.021.
8. Kitchenham, B. A., D. Budgen, and P. Brereton, *Evidence-based software engineering and systematic reviews*. 2016: CRC Press. Boca Raton, FL, USA. doi: 10.1201/b19467.
9. Budgen, D., P. Brereton, N. Williams, and S. Drummond, What Support do Systematic Reviews Provide for Evidence-informed Teaching about Software Engineering Practice? *e-Informatica Software Engineering Journal*, 2020. 14(1), p. 7-60. doi: 10.37190/e-Inf200101.
10. Ulziit, B., Z. A. Warraich, C. Gencel, and Kai Petersen, A conceptual framework of challenges and solutions for managing global software maintenance. *Journal of Software: Evolution and Process*, 2015. 27(10), p. 763-792. doi: 10.1002/smr.1720.
11. Benestad, H. C., B. Anda, and E. Arisholm, Understanding software maintenance and evolution by analyzing individual changes: a literature review. *Journal of Software Maintenance and Evolution: Research and Practice*, 2009. 21(6), p. 349-378. doi: 10.1002/smr.412.
12. Li, B., X. Sun, H. Leung, and S. Zhang, A survey of code-based change impact analysis techniques. *Software Testing, Verification and Reliability*, 2013. 23(8), p. 613-646. doi: 10.1002/stvr.1475.
13. Riaz, M., E. Mendes, and E. Tempero, A Systematic Review of Software Maintainability Prediction and Metrics. In *Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement*, 2009. p. 367-377. doi: 10.1109/ESEM.2009.5314233.
14. Sulman, S., and B. Nisar, A review of software fault detection and correction process, models and techniques. *Journal of Software Engineering & Intelligent Systems*, 2018. 3(1), p. 37-44.
15. Li, H., X. Xu, B. Jiang, J. Wei, and J. Wang. Modeling Software Systems as Complex Networks: Analysis and Their Applications. *Mathematical Problems in Engineering*, 2020. Article ID 5346498. doi: 10.1155/2020/5346498.
16. Savić, M., M. Ivanović, and L. C. Jain, *Complex Networks in Software, Knowledge, and Social Systems*. 2019: Springer. Cham, Switzerland. doi: 10.1007/978-3-319-91196-0.
17. Brereton, P., B.A. Kitchenham, D. Budgen, M. Turner, and M. Khalil, Lessons from applying the systematic literature review process within the software engineering domain. *Journal of Systems and Software*, 2007. 80(4), p. 571-583, doi: 10.1016/j.jss.2006.07.009.
18. O'Brien, M. P., J. Buckley, and T. M. Shaft., Expectation-based, inference-based, and bottom-up software comprehension. *Journal of software maintenance and evolution: Research and practice*, 2004. 16(6), p. 427-447. doi: 10.1002/smr.v16:6.
19. Sommerville, I., *Software Engineering, 9th ed.* 2011: Addison Wesley. Boston, MA, USA.

20. Mens, T., An Ecosystemic and Socio-Technical View on Software Maintenance and Evolution. In *Proceedings of 2016 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. 2016, p. 1-8, doi: 10.1109/ICSME.2016.19.
21. Liu, Z., Y-C. Lai, and N. Ye, Propagation and immunization of infection on general networks with both homogeneous and heterogeneous components. *Physical Review E*, 2003. 67(3), p. 031911. doi: 10.1103/PhysRevE.67.031911.
22. Rouse, W. B., and N. Serban, Complex Socio-Technical Systems - Understanding and Influencing Causality of Change. *Information Knowledge Systems Management*, 2011. 10(1-4), p. 25-49. doi: 10.3233/IKS-2012-0184.
23. Shull, F., J. Singer, and D.I.K.Sjøberg (Eds.), *Guide to Advanced Empirical Software Engineering. 1st ed.* 2008: Springer-Verlag London. London, UK.

AUTHORS PROFILES

Zeljko Stojanov received PhD degree in Computer science and applied informatics at the University of Novi Sad, Serbia. He works as an associate professor at the University of Novi Sad, Technical Faculty ”Mihajlo Pupin”, Serbia. He has over fifteen years of experience working with small software companies as a software developer and as a consultant in the fields of software development, software maintenance and process improvement. His research interests are in the fields of software engineering, process assessment and improvement, business informatics, learning and knowledge management, engineering education, and human aspects of engineering. He is author of scientific papers published in refereed journals and in the proceedings of international conferences. He participated in several research and industrial projects at national and international levels. He is the member of IEEE and ACM.

Jelena Stojanov received PhD degree in Mathematics at the University of Novi Sad, Serbia. She works as an assistant professor at the University of Novi Sad, Technical Faculty ”Mihajlo Pupin”, Serbia. Her research interests are in the fields of applied mathematics, graph theory and complex networks, statistics and differential geometry. She published scientific articles in refereed journals and in the proceedings of international conferences. She participated in several research and development projects at national and international levels.