Situational Factors for Modern Code Review to Support Software Engineers' Sustainability

Sumaira Nazir[1], Nargis Fatima[2], Suriayati Chuprat[3]
Razak Faculty of Technology and Informatics
University Technology Malaysia (UTM), Kuala Lumpur, Malaysia[1, 2, 3]
Department of Engineering and Computer Science
National University of Modern Languages (NUML), Islamabad, Pakistan[1, 2]

***Abstract*—Software engineers working in Modern Code Review (MCR) are confronted with the issue of lack of competency in the identification of situational factors. MCR is a software engineering activity for the identification and fixation of defects before the delivery of the software product. This issue can be a threat to the individual sustainability of software engineers and it can be addressed by situational awareness. Therefore, the objective of the study is to identify situational factors concerning the MCR process. Systematic Literature Review (SLR) has been used to identify situational factors. Data coding along with continuous comparison and memoing procedures of grounded theory and expert review has been used to produce an exclusive and validated list of situational factors grouped under categories. The study results conveyed 23 situational factors that are grouped into 5 broad categories i.e. People, Organization, Technology, Source Code and Project. The study is valuable for researchers to extend the research and for software engineers to identify situations and sustain for longer.**

Keywords—Situational; modern code review; sustainable software engineer

I. Introduction

Sustainable software engineering is presently a major concern in software development [1], [2]. It has five major aspects such as individual, social, economic, environmental, and technical [3], [4]. It is argued that work has been done regarding technical, economic, social, and environmental aspect of sustainable software engineering, however, individual sustainability aspect has been given less attention by the researchers and it warrants future research [2], [3], [4], [5], [6], [7].

Regarding individual sustainability, the software engineers are confronting with the issue of lack of competency in the identification of situational factors in various software engineering activities such as software requirement gathering and design, software construction and testing, modern code review [3], [8], [9].

The existing work concerning identification of situational factors have got attention by the researcher in software requirement and for software development, however, less attention has been devoted concerning situational context in modern code review specifically, to support software engineers' sustainability [3], [8], [9], [10], [11], [12] that can be the reason of software failure [3], [11], [13], [14].

Therefore, there is a need to identify situational factors for Modern Code Review (MCR) to overcome the issue of lack of competency in the identification of situational factors and to ensure the software engineers' sustainability involved in the MCR process [8], [9], [15]. MCR is an enhancement of Fagan's inspection, commonly known as a lightweight review process [16], [17]. It is supposed as a significant tool for improving code and software quality [16], [17], [18]. In MCR, software engineers i.e. authors and reviewers both aimed to improve the code quality [16], [17]. In this process, the code is reviewed by the reviewer from varying aspects. For instance, code style, code logic, code complexity etc. [16], [17], [18]. The process is highly reliant on review tools such as Code flow, Review board, Gerrit, etc. [16], [19].

This research has twofold aims i.e. to perform Systematic literature review (SLR) to identify situational factors for the MCR process and to validate them through expert review. The study detailed the SLR phases and expert review to identify and validate the situational factors for the MCR process that can help software engineers to sustain longer. This inquiry at one hand allows the investigators to outspread the research and on the other hand, it supports software engineers' sustainability through situational awareness in MCR.

The paper is arranged as Section II provides the background details. Section III details the study methodology. Section IV presents the results. Section V delivers the discussion. Section VI provides the conclusion and future work. Section VII deliberates the research contributions.

II. Background

Sustainability in software engineering is a noteworthy part of practices in the disciplines [1], [2]. It is defined as the "capacity to endure" [4]. There are five sustainability aspects reported in the literature such as individual, social, economic, environmental and technical [3], [4], [13], [14].

Economic sustainability aspects deals with investments and profitability [3], [4], [20], [21]. The technical sustainability aspect is connected to the ability to maintain and evolve software [8], [22], [23]. The social sustainability aspect is associated with the relationship between organizations, groups, and individuals [3], [4]. The environmental sustainability aspect is related to the objective to lessen the negative influence of software engineering activities on the environment [2], [3], [24], [25]. The individual sustainability aspect is related to well-being, education, and liberty of software engineers to sustain for longer [3], [4], [14]. Although valued work has been performed concerning to sustainable software engineering aspects i.e. technical, economic, social, and environmental, however, individual sustainability aspect has got less attention by the researchers and needs a detailed insight [2], [3], [4], [5], [6], [7].

Presently, software engineers are confronted with the issue concerning their sustainability such as lack of competency in the identification of situational factors in the Modern Code Review (MCR) process [3], [8], [9], [13], [14]. MCR is an important software engineering activity also known as a lightweight version of Fagan's inspection where developer other than source code author review the source code and guide the author in improving the quality of source code [16], [17], [26], [27], [28]. The process is performed with the help of review tools, for instance, Gerrit [16]. The overview of the process is shown in Fig. 1.

One of the reasons behind the issue of lack of competency in the identification of situational factors is unfamiliar situations [9], [23]. It is stated that the above issue can cause a decrease in the competency and capability of software engineers towards problem understanding, and identification of unfamiliar situations [9], [15]. This issue can be addressed by situational awareness in MCR [9], [11], [14].

In modern software development, situation-aware computing is extremely desirable [9]. Situation aware software engineering also called situational software engineering allows the software engineers to be able to deal with familiar situations instead of being unproductive with unfamiliar situations [9], [10], [11], [15], [29]. Situational software engineering ensures the software engineers' sustainability [9], [11], [15].

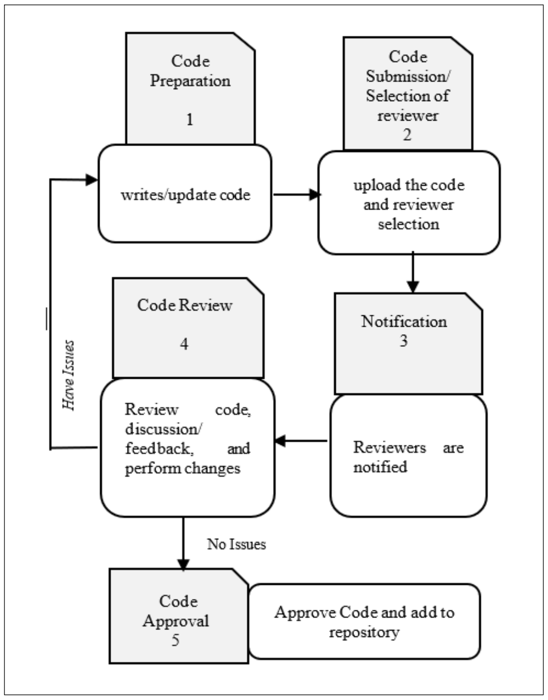
Code Preparation 1
writes/update code
Code Submission/ Selection of reviewer 2
upload the code and reviewer selection
Notification 3
Reviewers are notified
Code Review 4
Review code, discussion/ feedback, and perform changes
Have Issues
No Issues
Code Approval 5
Approve Code and add to repository

Fig. 1. MCR Overview [17].

The previous work established that researchers have highlighted the significance of situational factors identification and attention has been given on the identification of situational factors in software requirement and for software development [3], [8], [9], [10], [11], [12], however, little indication is available concerning to Modern Code Review (MCR) [30], [31], [32], [33], [34]. It results in the unavailability of situational guidelines that can help software engineers to increase their competency for the identification of situational factors. Therefore, this study aims to identify the unique and validated list of situational factors for the MCR process to support software engineers' sustainability.

III. Research Methodology

The research activities completed to produce the unique and validated list of situational factors for the MCR process to support software engineers' sustainability are explained in subsections.

A. Systematic Literature Review (SLR)

The Systematic Literature Review (SLR) technique specified by [35] has been employed to recognize the situational factors for the MCR process to support software engineers' sustainability. The SLR technique is a schematized approach to accomplish impartial results [35]. It involves significant steps such as SLR planning, SLR execution and, reporting the SLR results. It is a suitable technique to record significant data from existing research. The steps involved in SLR are explained in subsections.

1) Research questions: Designing the research question is an important aspect of SLR. For this study, the PICOC strategy proposed by [36] has been utilized to prepare the research question. PICOC stands for population, intervention, comparison, outcome, and context. As this study is inclusive of any type of comparison between methods, therefore the study selected only population, intervention, outcome, and context that is PIOC. Table I presents a summary of the PIOC strategy.

To collect the indications on the current state of research concerning situational factors, the planned research question is given below.

RQ1: What situational factors of the MCR process should be known, to support the software engineers' sustainability?

2) Search technique: The search technique includes the recognition of main search terms, finding their alternative and then constructing the search thread to search relevant data from data sources. Table II presents the main terms and their alternatives.

The search thread is planned based on the main terms and their alternate terms. The planned search thread is given below.

('conditional', 'contextual', 'situational factor') AND ('modern code inspection', 'contemporary code review' 'code review', 'code inspection', ' 'lightweight code inspection') AND ('sustainable software engineers', 'sustainable software developer', 'sustainable software programmer')

TABLE. I. POIC SUMMARY

Population	Intervention	Outcome	Context
Software Engineers	MCR Process	Situational factors for MCR process to support software engineers' sustainability	The research includes all types of studies, for instance, interviews, surveys, questionnaires, experiments, and case studies regarding MCR.

TABLE. II. MAIN TERMS AND THEIR ALTERNATES

Situational	Modern Code Review	Sustainable Software Engineer
'conditional', 'contextual', 'situational factor'	'modern code inspection', 'contemporary code review' 'code review', 'code inspection', 'lightweight code inspection'	'sustainable software developer', 'sustainable software programmer'

3) Data source: The data is collected from varying sources known for publishing software engineering research articles. The data origin utilized for the study includes ACM, IEEE, Springer link, Science direct, Wiley online, Scopus, and Web of Science. Papers published in 2013 to 2019 are considered for selection. The journals' articles, workshop papers, conference papers, book chapters, published thesis, and technical reports are searched in the defined databases. The data sources that reflect situational factors that impact the sustainability of software engineers involved in the MCR process are recognized as probably pertinent.

4) Study inclusion and exclusion principles: The inclusion criteria for including the relevant studies is as follows.

a) Research published in journals, conference proceedings, workshops, book chapters, thesis, or technical reports that are discussing situational factors for the MCR process to support software engineers.

b) Publication content is available completely.

c) Publications from 2013 to 2019.

d) Research papers are written in the English language.

The research papers were excluded based on the exclusion criteria specified below.

a) Research papers giving information such as conference proceedings, workshops table of content, and irrelevant title.

b) Research papers that do not contain any one of the study main terms or their alternates terms.

c) Duplicate research papers

Fig. 2 summarizes the study inclusion and exclusion principles.

5) Quality assessment: The selected research papers are weighed for their quality by using the checklist provided by [35]. The checklist used for evaluating the quality of the research papers is given in Table III. Furthermore, each question given in the checklist presented in Table III is answered by the measures given by [35]. The measures are shown in Table IV.

6) Data extraction: The data is extracted from the selected studies with the help of extraction forms given by [35]. The details about the data extraction form are presented in Table V.

TABLE. III. QUALITY ASSESSMENT CHECKLIST [36]

Question	Answer
Are the objective clearly stated?	Yes/ No/Partially
Are the findings sound and significant?	-
Are the prediction techniques used clearly described and their selection are justified?	-
Is the facts been extended through the research?	-
Is the multiplicity of viewpoint and background been explored?	-
Are the associations between data, interpretation, and conclusions are vibrant?	-
Does the depth of the data is conveyed?	-

TABLE. IV. MEASURES FOR ANSWERING QUESTION GIVEN IN CHECKLIST [36]

Answer	Score
Yes	1
No	0
Partially	0.5

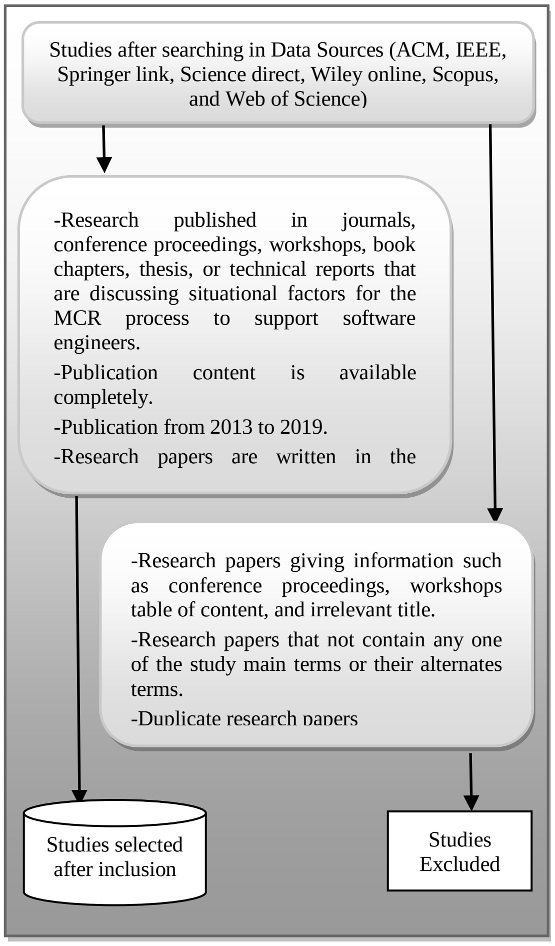
Studies after searching in Data Sources (ACM, IEEE, Springer link, Science direct, Wiley online, Scopus, and Web of Science)
-Research published in journals, conference proceedings, workshops, book chapters, thesis, or technical reports that are discussing situational factors for the MCR process to support software engineers.
-Publication content is available completely.
-Publication from 2013 to 2019.
-Research papers are written in the
-Research papers giving information such as conference proceedings, workshops table of content, and irrelevant title.
-Research papers that not contain any one of the study main terms or their alternates terms.
-Duplicate research papers
Studies selected after inclusion
Studies Excluded

Fig. 2. Study Inclusion and Exclusion Criteria.

TABLE. V. DATA EXTRACTION FORM

Data Items	Data information	Notes
Paper ID	A unique identifier SFS<1--n>	
Title		
Author (s)		
Year		
Study type	(Conference/Journal/Book/Thesis)	
Study Publisher	IEEE	
Situational Factors		

7) Data analysis: After the data extraction, qualitative data analysis has been performed using the grounded theory given by [37], [38]. The grounded theory techniques i.e. data coding, continuous comparison and memoing has been used to recognize a unique list of situational factors grouped under various categories for the MCR process.

B. Expert Review

After finding the unique list of situational factors grouped under various categories, the list has been sent to the experts for the assessment concerning naming, terms, and classification of identified situational factors. The experts are also asked to mention new situational factors. The experts are designated based on their expertise in MCR along with software development experience for more than 10 years, the familiarity of situational software engineering, sustainable software engineering, and individual sustainability. The guidelines given by Ayyub [39], [40] are followed. The final list of situational factors along with their classification is presented in Section VI. Fig. 3 highlights the summarized view of the methodology employed for the identification of situational factors for the MCR process to support software engineers' sustainability.

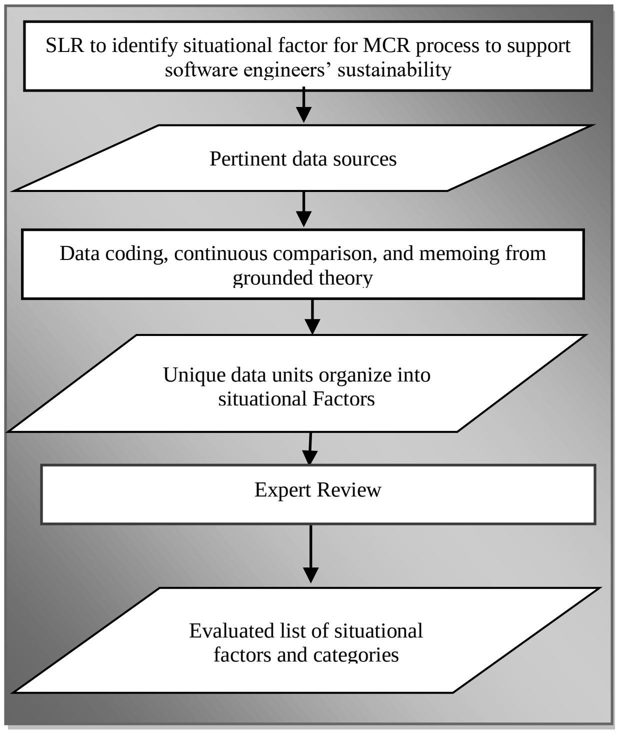
SLR to identify situational factor for MCR process to support software engineers' sustainability
Pertinent data sources
Data coding, continuous comparison, and memoing from grounded theory
Unique data units organize into situational Factors
Expert Review
Evaluated list of situational factors and categories

Fig. 3. Overview of Study Methodology.

IV. RESULTS

This section presents the results of SLR and the expert review i.e. the study selection process, distribution of data sources and an evaluated list of situational factors for the MCR process to support software engineers' sustainability.

A. Study Selection Process Results

In the initial search, 9295 papers are found based on defined study main terms. The studies mentioning exclusive information concerning table of content, workshop or conference preceding details or having disparate titles are eliminated, and 1096 studies are selected. The 1096 studies are assessed for relevancy concerning main terms of the study i.e. Situational, Modern Code Review, Sustainable Software Engineer. The studies not representing any of the study main terms are rejected and 187 studies are included. Afterward, 187 obtained studies are checked for replication. After replication assessment 162 studies are obtained for their quality assessment. After quality assessment 158 studies are included for deep review for the identification of situational factors for the MCR process to support software engineers' sustainability.

B. Distribution of Data Sources

Total 9295 papers obtained after an initial search from defined databases. Finally, 158 papers are selected after going through inclusion/exclusion and quality assessment.

C. Situational Factors

The study results reported 23 situational factors grouped into five categories i.e. people, organization, project, source code, and technology. The classification of the identified situational factors is discussed in subsections. Table VI summarizes the situational factors along with their classification for the MCR process to support software engineers' sustainability.

1) People: This category includes factors that are directly related to people. The situational factors included in this category are team, team interaction, reviewer response, and knowledge sharing [17], [19], [26], [34], [41], [42], [43], [44], [45], [46], [47], [48].

2) Source code: It refers to a list of human-readable instructions that a programmer writes using code editors. The source code runs through a compiler to turn it into machine code, that a computer can understand and execute [49]. The situational factors grouped under this category are source code attributes, source code change attributes, source code change documentation, testing, review concentration, and defect [17], [18], [19], [32], [43] [50], [51], [52], [53], [54], [55].

3) Organization: It refers to the group of people and facilities with an arrangement of tasks, authorities, and relations [56]. The situational factors included in this category are resources [26], organization policy, organization practices, organization standards, organization attributes, and information dissemination [26], [41], [43], [50], [57], [58].

4) Project: The project is an arrangement of tasks that are prearranged from beginning to end bounded by resources and required outcomes [56]. This category involves two situational factors i.e. project attributes and project release management [16], [17], [43], [59], [60].

TABLE. VI. SITUATIONAL FACTORS AND THEIR CATEGORIES

Categories	Situational Factors	References
People	Team	[17], [19], [26], [34], [41], [42], [43], [65], [66], [61], [67], [50], [68]
	Team Interaction	[17], [26], [42], [43], [65], [50], [69], [70],[59], [71], [72]
	Reviewer Response	[16], [17], [18], [26], [32], [42], [43], [66], [44], [45], [73]
	Knowledge Sharing	[26], [43], [61], [67], [69], [46], [47]
Source Code	Source Code Attributes	[17], [18], [19], [30], [31], [32], [33], [41] [42], [43], [66], [70], [59], [44], [47] [74], [51] , [52], [75], [76], [77], [66]
	Source Code Change Attributes	[16], [17], [18], [19] [26], [30], [32], [43] [50], [70], [59], [71] [51], [52], [75], [78]
	Source Code Change Documentation	[16], [18], [19], [26], [30], [42], [43], [65] [50], [59], [51], [76], [79]
	Testing	[18], [19], [50], [51], [52], [53], [54]
	Review Concentration	[17], [32], [42], [55]
	Defect	[32], [41], [70], [71], [55], [62]
Organization	Resources	[26]
	Organization Policy	[26], [57]
	Organization Practices	[26], [41], [43], [50], [57], [58]
	Organization Standards	[17], [18], [26], [41], [61], [57]
	Organization Attributes	[17], [26], [41], [61]
	Information Dissemination	[57]
Project	Project Attributes	[16], [17], [43], [59]
	Project Release Management	[17], [18], [43],
Technology	Process	[17], [18], [26], [31], [34] [41] [43], [61], [71], [57]
	Tool	[18], [26] [34], [61], [50], [69], [70], [59], [71], [74], [62], [63], [64]
	Technology Maturity	[26]
	Technology Accessibility	[26], [61]
	Training	[17], [18], [26], [70]

5) Technology: It refers to the approaches, skills, and processes used in the creation of goods or services in the achievement of aims [49]. The situational factors included in this category are process, tool, technology maturity, technology accessibility, and training [18], [26], [31], [34] [41] [43], [61], [62], [63], [64].

V. DISCUSSION

This study has provided a comprehensive list of classified and validated situational factors for the MCR process to support software engineers' sustainability through SLR and expert review. The identified situational factors that can impact the sustainability of software engineers can be an important reference for researchers involved in research concerning situational software engineering, sustainable software engineering, and MCR. The work can support the sustainability of software engineers involved in the MCR process by providing the list of situational factors. The identified list can also act as a guide for the researchers and practitioners working in situational software engineering, and sustainable software engineering.

The study presents the situational factors based on literature. Although effort has been made to cover all the related research papers to present the comprehensive list of situational factors for MCR process to support software engineers' sustainability, however, there can be a possibility that some research may not be covered.

VI. CONCLUSION AND FUTURE WORK

This work has provided a unique, classified, and validated list of situational factors for the MCR process to support software engineers' sustainability. A total of 23 situational factors have been identified as a result of this work. The identified factors are broadly grouped under five categories i.e. People, Organization, Process, Source code, and Technology. These factors can support the sustainability of software engineers.

In the future, a more inclusive list will be shaped, the ongoing research objectives. In addition to this, a comprehensive and enhanced MCR process will be produced with situational factors. This work provided situational factors for MCR activity of software engineering that allows the investigators to extend this research by determining other situational factors in other software engineering activities.

VII. CONTRIBUTION

The examination contributed to the software engineering body of knowledge (SWEBOK), Situational software engineering and sustainable software engineering by highlighting the worth of situational factor identification for the sustainability of the software engineers involved in MCR.

REFERENCES

[1] B. Penzenstadler and H. Femmer, "Towards a Definition of Sustainability in and for Software Engineering at Sustainable System : Product," p. 2013, 2013.

[2] S. Naumann, E. Kern, M. Dick, and T. Johann, "Sustainable Software Engineering: Process and Quality Models, Life Cycle, and Social Aspects," ICT Innov. Sustain. Adv. Intell. Syst. Comput., vol. 310, pp. 191–205, 2015.

[3] R. Chitchyan, I. Groher, and J. Noppen, "Uncovering sustainability concerns in software product lines," J. Softw. Evol. Process, vol. 29, no. 2, pp. 1–20, 2017.

[4] B. Penzenstadler et al., “Software Engineering for Sustainability: Find the Leverage Points!,” IEEE Softw., vol. 35, no. 4, pp. 22–33, 2018.

[5] S. Naumann, M. Dick, E. Kern, and T. Johann, “The GREENSOFT Model: A reference model for green and sustainable software and its engineering,” Sustain. Comput. Informatics Syst., vol. 1, no. 4, pp. 294–304, 2011.

[6] A. D. Komeil Raisian, Jamaiah Yahaya, “Sustainable Software Development Life Cycle Process Model Based on Capability Maturity Model Integration : a Study in Malaysia,” J. Threoretical Appl. Inf. Technol., vol. 95, no. 21, pp. 5723–5734, 2017.

[7] B. Penzenstadler, A. Raturi, D. Richardson, C. Calero, H. Femmer, and X. Franch, “Systematic Mapping Study on Software Engineering for Sustainability (SE4S),” in Proc. 18th International Conference on Evaluation and Assessment in Software Engineering, 2014, pp. 1–14.

[8] R. Chitchyan, L. Duboc, C. Becker, S. Betz, B. Penzenstadler, and C. C. Venters, “Sustainability Design in Requirements Engineering : State of Practice,” in IEEE/ACM 38th IEEE International Conference on Software Engineering, 2016, pp. 533–542.

[9] A. A. Abbood and G. Sulong, “Segmentation and Enhancement of Fingerprint,” Springer Int. Publ., vol. 5, no. 3, 2018.

[10] P. Clarke, R. V. O. Connor, R. V. O. Connor, and B. Leavy, “A complexity theory viewpoint on the software development process and situational context,” no. May, 2016.

[11] H. H. Khan and M. N. Malik, “Software Standards and Software Failures: A Review with the Perspective of Varying Situational Contexts,” IEEE Access, vol. 5, pp. 17501–17513, 2017.

[12] P. Clarke and R. V. O’Connor, “The situational factors that affect the software development process: Towards a comprehensive reference framework,” Inf. Softw. Technol., vol. 54, no. 5, pp. 433–447, 2012.

[13] S.Nazir, N. Fatima, and S. Chuprat “Individual Sustainability Barriers and Mitigation Strategies : Systematic Literature Review Protocol.”

[14] S. Nazir, N. Fatima, S. Chuprat, H. Sarkan, N. F. Nilam, and N.A. Sajarif, “Sustainable Software Engineering:A Perspective of Individual Sustainability,” Int. J. Adv. Sci. Eng. Inf. Technol.

[15] G. Marks, R. V. O’Connor, and P. M. Clarke, “The impact of situational context on the software development process – A case study of a highly innovative start-up organization,” Commun. Comput. Inf. Sci., vol. 770, pp. 455–466, 2017.

[16] A. Bacchelli and C. Bird, “Expectations, outcomes, and challenges of modern code review,” in Proc. International Conference on Software Engineering, 2013, pp. 712–721.

[17] A. Bosu, J. C. Carver, C. Bird, J. Orbeck, and C. Chockley, “Process Aspects and Social Dynamics of Contemporary Code Review: Insights from Open Source Development and Industrial Practice at Microsoft,” IEEE Trans. Softw. Eng., vol. 43, no. 1, pp. 56–75, 2017.

[18] O. Kononenko, O. Baysal, and M. W. Godfrey, “Code Review Quality: How Developers See It,” in Proc. International Conference on Software Engineering, 2016, pp. 1028–1038.

[19] A. Ram, Achyudh ; Sawant, Anand; Castelluccio, Marco; Bacchelli, “What Makes a Code Change Easier to Review? An Empirical Investigation on Code Change Reviewability,” in Proc. ESEC/FSE, 2018.

[20] M. L. Gibson, C. C. Venters, M. Palacin-silva, and N. Seyff, “Mind the chasm: A UK fisheye lens view of sustainable software engineering,” 2017.

[21] R. Ahmad, F. Baharom, and A. Hussain, “Software Sustainability Development : Impactibility Characteristic Focuses on Social Approach,” in Proc. 6th International Conference on Computing and Informatics, 2017, no. 093, pp. 595–600.

[22] H. Koziolek, “Sustainability Evaluation of Software Architectures : A Systematic Review,” in Proc. QoSA+ISARCS, 2011, pp. 3–12.

[23] C. Becker et al., “Requirements: The key to sustainability,” IEEE Softw., vol. 33, no. 1, pp. 56–65, 2016.

[24] E. Kern et al., “Sustainable software products—Towards assessment criteria for resource and energy efficiency,” Futur. Gener. Comput. Syst., vol. 86, no. 3715, pp. 199–210, 2018.

[25] I. Manotas et al., “An empirical study of practitioners’ perspectives on green software engineering,” in IEEE/ACM 38th IEEE International Conference on Software Engineering, 2016, pp. 237–248.

[26] L. MacLeod, M. Greiler, M. A. Storey, C. Bird, and J. Czerwonka, “Code Reviewing in the Trenches: Challenges and Best Practices,” IEEE Softw., vol. 35, no. 4, pp. 34–42, 2018.

[27] N. Fatima, S. Chuprat, and S. Nazir, “Challenges and Benefits of Modern Code Review-Systematic Literature Review Protocol,” in Proc. International Conference on Smart Computing and Electronic Enterprise, 2018, pp. 1–5.

[28] S. Nazir, N. Fatima, and S. Chuprat, “Modern Code Review Benefits–Primary findings of a systematic literature review,” in The 3rd International Conference on Software Engineering and Information Management.

[29] C. K. Chang, “Situation Analytics: A Foundation for a New Software Engineering Paradigm,” Computer (Long. Beach. Calif)., vol. 49, no. 1, pp. 24–33, 2016.

[30] E. W. dos Santos and I. Nunes, “Investigating the Effectiveness of Peer Code Review in Distributed Software Development,” in Proc. 31st Brazilian Symposium on Software Engineering, 2017, pp. 84–93.

[31] T. Baum, F. Kortum, K. Schneider, A. Brack, and J. Schauder, “Comparing pre-commit reviews and post-commit reviews using process simulation,” J. Softw. Evol. Process, vol. 29, no. 11, pp. 1–15, 2017.

[32] F. Armstrong, F. Khomh, and B. Adams, “Broadcast vs. Unicast Review Technology: Does It Matter?,” in Proc. 10th IEEE International Conference on Software Testing, Verification and Validation, 2017, pp. 219–229.

[33] F. Ebert, F. Castor, N. Novielli, and A. Serebrenik, “Confusion in Code Reviews: Reasons, Impacts, and Coping Strategies,” SANER 2019 - Proc. 2019 IEEE 26th Int. Conf. Softw. Anal. Evol. Reengineering, pp. 49–60, 2019.

[34] C. Sadowski, E. Söderberg, L. Church, M. Sipko, and A. Bacchelli, “Modern code review: : A Case Study at Google,” in Proc. ACM/IEEE 40th International Conference on Software Engineering: Software Engineering in Practice, 2018, pp. 181–190.

[35] B. Kitchenham and S. Charters, “Source: " Guidelines for performing Systematic Literature Reviews in SE " , Kitchenham et al Guidelines for performing Systematic Literature Reviews in Software Engineering Source: " Guidelines for performing Systematic Literature Reviews i,” pp. 1–44, 2007.

[36] F. Terms, “ M. Petticrew and H. Roberts. Systematic Reviews in the Social Sciences: A Practical Guide . Oxford: Blackwell 2006. 352 pp. ISBN 1 4051 2110 6. £29.99 ,” Couns. Psychother. Res., vol. 6, no. 4, pp. 304–305, 2006.

[37] Kathy Charmaz, Constructing Grounded Theory, A practical Guide through Qualitative Analysis. 2007.

[38] K.-J. Stol, P. Ralph, and B. Fitzgerald, “Grounded theory in software engineering research,” no. October 2017, pp. 120–131, 2016.

[39] B. Ayyub, “A practical guide on conducting expert-opinion elicitation of probabilities and consequences for corps facilities,” Inst. Water Resour. Alexandria, VA, USA, no. January, 2001.

[40] R. Boring, D. Gertman, J. Joe, and J. Marble, “Simplified expert elicitation guideline for risk assessment of operating events,” … Natl. Lab. INL …, no. June, p. 65, 2005.

[41] T. Baum, O. Liskin, K. Niklas, and K. Schneider, “Factors influencing code review processes in industry,” Proc. 2016 24th ACM SIGSOFT Int. Symp. Found. Softw. Eng. - FSE 2016, pp. 85–96, 2016.

[42] M. Di Biase, M. Bruntink, and A. Bacchelli, “A security perspective on code review: The case of chromium,” in Proc. IEEE 16th International Working Conference on Source Code Analysis and Manipulation, 2016, pp. 21–30.

[43] O. Kononenko, T. Rose, O. Baysal, M. Godfrey, D. Theisen, and B. De Water, “Studying Pull Request Merges : A Case Study of Shopify ’ s Active Merchant,” in Proc. 40th International Conference on Software Engineering: Software Engineering in Practice, 2018, pp. 124–133.

[44] S. McIntosh and Y. Kamei, “Are Fix-Inducing Changes a Moving Target? A Longitudinal Case Study of Just-In-Time Defect Prediction,” IEEE Trans. Softw. Eng., vol. 44, no. 5, pp. 412–428, 2018.

[45] S. Ruangwan, P. Thongtanunam, A. Ihara, and K. Matsumoto, “The Impact of Human Factors on the Participation Decision of Reviewers in Modern Code Review,” Empir. Softw. Eng. Manuscr., pp. 1–43, 2018.

[46] C. Parnin et al., "The Top 10 Adages in Continuous Deployment," IEEE Softw., vol. 34, no. 3, pp. 86–95, 2017.

[47] X. Yang, R. G. Kula, N. Yoshida, and H. Iida, "Mining Code Review Repositories : People , Process and Product," IEEE Work. Conf. Min. Softw. Repos., pp. 16–19, 2016.

[48] S. Nazir, N. Fatima, and S. Chuprat, "Situational factors affecting Software Engineers Sustainability: A Vision of Modern Code Review," in 6th IEEE International Conference on Engineering Technologies and Applied Sciences (ICETAS) , in press.

[49] ISO/IEC and IEEE, "ISO/IEC/IEEE 24765:2010 - Systems and software engineering -- Vocabulary," Iso/Iec Ieee, vol. 2010, p. 410, 2010.

[50] G. Gousios, M.-A. Storey, and A. Bacchelli, "Work practices and challenges in pull-based development," in Proc. 38th International Conference on Software Engineering, 2016, pp. 285–296.

[51] S. Fakhoury, "The Effect of Poor Source Code Lexicon and Readability on Developers´s Cognitive Load," in Proc. ICPC, 2018, pp. 286–296.

[52] D. Spadini, A. Bacchelli, M. Bruntink, F. Palomba, and L. Pascarella, "Information Needs in Contemporary Code Review," in Proc. ACM on Human-Computer Interaction, 2018, vol. 2, no. CSCW, pp. 1–27.

[53] J. Tsay, L. Dabbish, and J. Herbsleb, "Let's talk about it: evaluating contributions through discussion in GitHub," in Proc. 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering - FSE 2014, 2014, pp. 144–154.

[54] Y. Yu, H. Wang, V. Filkov, P. Devanbu, and B. Vasilescu, "Wait for It: Determinants of pull request evaluation latency on GitHub," in Proc. IEEE International Working Conference on Mining Software Repositories, 2015, vol. 2015-Augus, pp. 367–371.

[55] C. Thompson and D. Wagner, "A Large-Scale Study of Modern Code Review and Security in Open Source Projects," Proc. 13th Int. Conf. Predict. Model. Data Anal. Softw. Eng., pp. 83–92, 2017.

[56] I. Standard, ISO/IEC/IEEE Systems and software engineering -- System life cycle processes, vol. 8. 2013.

[57] D. M. German, U. Rey, and J. Carlos, "' Was my contribution fairly reviewed ?' A Framework to Study the Perception of Fairness in Modern Code Reviews," in Proc. ACM/IEEE 40th International Conference on Software Engineering Synthesizing, 2018, no. 2, pp. 523–534.

[58] M. Barnett, C. Bird, J. Brunet, and S. K. Lahiri, "Helping developers help themselves: Automatic decomposition of code review changesets," Proc. - Int. Conf. Softw. Eng., vol. 1, no. August 2014, pp. 134–144, 2015.

[59] P. C. Rigby and C. Bird, "Convergent Contemporary Software Peer Review Practices Categories and Subject Descriptors," in Proc. ESEC/FSE, 2013, pp. 202–212.

[60] S. Nazir, N. Fatima, and S. Chuprat, "Does Project Associated Situational Factors have Impact on Sustainability of Modern Code Review Workforce?," in 6th IEEE International Conference on Engineering Technologies and Applied Sciences (ICETAS) , in press.

[61] A. Kalyan, M. Chiam, J. Sun, and S. Manoharan, "A Collaborative Code Review Platform for GitHub," in Proc. IEEE International Conference on Engineering of Complex Computer Systems, 2017, pp. 191–196.

[62] D. Singh, V. R. Sekar, K. T. Stolee, and B. Johnson, "Evaluating how static analysis tools can reduce code review effort," IEEE Symp. Vis. Lang. Human-Centric Comput., pp. 101–105, 2017.

[63] Z. X. Li, Y. Yu, G. Yin, T. Wang, and H. M. Wang, "What Are They Talking About? Analyzing Code Reviews in Pull-Based Development Model," J. Comput. Sci. Technol., vol. 32, no. 6, pp. 1060–1075, 2017.

[64] Y. Yu, H. Wang, G. Yin, and T. Wang, "Reviewer recommendation for pull-requests in GitHub: What can we learn from code review and bug assignment?," Inf. Softw. Technol., vol. 000, pp. 1–15, 2015.

[65] A. Bosu, J. Carver, R. Guadagno, B. Bassett, D. McCallum, and L. Hochstein, "Peer impressions in open source organizations: A survey," J. Syst. Softw., vol. 94, pp. 4–15, 2014.

[66] A. Ouni, R. G. Kula, and K. Inoue, "Search-based peer reviewers recommendation in modern code review," in Proc. - IEEE International Conference on Software Maintenance and Evolution, 2017, pp. 367–377.

[67] N. Kitagawa, H. Hata, A. Ihara, K. Kogiso, and K. Matsumoto, "Code Review Participation: Game Theoretical Modeling of Reviewers in Gerrit Datasets," in Proc. 9th International Workshop on Cooperative and Human Aspects of Software Engineering, 2016, pp. 64–67.

[68] C. Bird, T. Carnahan, and M. Greiler, "Lessons learned from building and deploying a code review analytics platform," IEEE Int. Work. Conf. Min. Softw. Repos., pp. 191–201, 2015.

[69] T. Baum, O. Liskin, K. Niklas, and K. Schneider, "A Faceted Classification Scheme for Change-Based Industrial Code Review Processes," Proc. - 2016 IEEE Int. Conf. Softw. Qual. Reliab. Secur. QRS 2016, pp. 74–85, 2016.

[70] F. Ebert, F. Castor, N. Novielli, and A. Serebrenik, "Confusion detection in code reviews," in Proc. IEEE International Conference on Software Maintenance and Evolution, 2017, pp. 549–553.

[71] H. Lal and G. Pahwa, "Code review analysis of software system using machine learning techniques," in in Proc. 11th International Conference on Intelligent Systems and Control, 2017, pp. 8–13.

[72] F. Ebert, F. Castor, N. Novielli, and A. Serebrenik, "Communicative Intention in Code Review Questions," 2018.

[73] P. Thongtanunam, S. McIntosh, A. E. Hassan, and H. Iida, "Investigating code review practices in defective files: An empirical study of the Qt system," IEEE Int. Work. Conf. Min. Softw. Repos., vol. 2015-Augus, pp. 168–179, 2015.

[74] J. Kim and E. Lee, "Understanding review expertise of developers: A reviewer recommendation approach based on latent Dirichlet allocation," Symmetry (Basel)., vol. 10, no. 4, pp. 5–7, 2018.

[75] A. Luxton-reilly, A. Lewis, and B. Plimmer, "Comparing Sequential and Parallel Code Review Techniques for Formative Feedback," in Proc. 20th Australasian Computing Education Conference, 2018, pp. 45–52.

[76] A. Bosu, M. Greiler, and C. Bird, "Characteristics of useful code reviews: An empirical study at Microsoft," in Proc. IEEE International Working Conference on Mining Software Repositories, 2015, vol. 2015-Augus, pp. 146–156.

[77] Z. Xia, H. Sun, J. Jiang, X. Wang, and X. Liu, "A Hybrid Approach to Code Reviewer Recommendation with Collaborative Filtering," in SoftwareMining 2017, Urbana-Champaign, IL, USA, 2017, pp. 24–31.

[78] M. Beller, A. Bacchelli, A. Zaidman, and E. Juergens, "Modern code reviews in open-source projects: which problems do they fix?," in Proc. 11th Working Conference on Mining Software Repositories, 2014, pp. 202–211.

[79] O. Kononenko, O. Baysal, L. Guerrouj, Y. Cao, and M. W. Godfrey, "Investigating code review quality: Do people and participation matter?," in Proc. IEEE 31st International Conference on Software Maintenance and Evolution, 2015, pp. 111–120.