MODERN CODE REVIEW KNOWLEDGE SHARING MODEL TO REDUCE SOFTWARE ENGINEERING WAITING WASTE

NARGIS FATIMA

UNIVERSITI TEKNOLOGI MALAYSIA

MODERN CODE REVIEW KNOWLEDGE SHARING MODEL TO REDUCE SOFTWARE ENGINEERING WAITING WASTE

NARGIS FATIMA

A thesis submitted in fulfilment of the
requirements for the award of the degree of
Doctor of Philosophy

Razak Faculty of Technology and Informatics
Universiti Teknologi Malaysia

JULY 2021

DEDICATION

This thesis is dedicated to my father, whose unconditional love and encouragement sustained me throughout this Ph.D. Journey. It is also dedicated to my late mother who taught me to be patient and have faith in Allah in every situation.

ACKNOWLEDGEMENT

This research journey has been challenging. I thank Allah Tallah for providing me the courage and support to complete my research. To prepare this thesis many people, researchers, academicians, and practitioners guided and helped me in different ways. I would like to express my sincere gratitude to my supervisor Prof. Dr. Suriayati Chuprat for continuous support, encouragement, and immense knowledge. Her guidance helps me all the time in the research and writing of the thesis. As a learner, I cannot imagine a better advisor than her. Without her guidance and support, this research would not be possible. Besides my supervisor, I would like to thank the rest of my thesis committee for the insightful feedback that guided in to broaden my research. I am appreciative of Universiti Teknologi Malaysia (UTM) for partially funding my Ph.D. research and providing me an excellent research environment. I am also thankful to all my friends and colleagues, who supported me in the completion of my research work. Special thanks to my best friend Sumaira Nazir and her family for their continuous support. It is hard to explain how grateful I am for having such a friend like her. Finally, I would like to thank my family for their support to complete my Ph.D. research. I am especially thankful to my brother Syed Ammar Yasir and sister Surraiya Tabassum who supported me throughout my Ph.D. Journey.

ABSTRACT

Reducing waiting waste in software engineering activities such as software requirement gathering, software modelling and construction, software inspections, and modern code review is challenging. Waiting waste creates a blocking state for other tasks, delays project, decreases developers' productivity, and increases mental distress. One of the major causes of waiting waste generation is a lack of knowledge sharing in Modern Code Review (MCR). Although past studies have focused on knowledge sharing in other software engineering activities, little evidence is available in the context of MCR, resulting in the lack of knowledge sharing guidelines in MCR to guide software engineers to reduce software engineering waiting waste. This study developed a modern code review knowledge sharing model to reduce software engineering waiting waste. To develop the model, the knowledge sharing factors in MCR and the ranked most influential knowledge sharing factors for MCR activities were identified. A systematic literature review was conducted to identify the knowledge sharing factors, subfactors, and categories in MCR. An electronic knowledge sharing MCR guideline was also developed based on the MCR knowledge sharing model. Four software engineering experts validated the identified list of knowledge sharing factors, sub-factors, and categories in MCR for their naming conventions, grouping, and sub-grouping. A Delphi survey involving ten experts was employed to identify the most influential knowledge sharing factors for MCR activities. The results from the Delphi survey were used to develop the MCR knowledge sharing model. The relationships between the categories of the MCR knowledge sharing model - Individual, Team, Facility Conditions, Artefact, and Social - were explored using regression analysis. An electronic reference guide of the MCR knowledge sharing model was developed using ASP.NET and SQL server based on the developed MCR knowledge sharing model. The experiment was conducted with the support of the electronic reference guide of the MCR knowledge sharing model to evaluate the effectiveness of the developed model to reduce software engineering waiting waste. In sum, this study has developed MCR knowledge sharing mode, which constitutes of evaluated list of knowledge sharing factors in MCR, and the most influential knowledge sharing factors for MCR activities to reduce software engineering waiting waste.

ABSTRAK

Mengurangkan pembaziran menunggu dalam aktiviti kejuruteraan perisian seperti pengumpulan keperluan perisian, permodelan dan pembangunan perisian, pengujian perisian, serta tinjauan kod moden merupakan sesuatu yang mencabar. Pembaziran menunggu mewujudkan keadaan terhalang bagi tugas-tugas lain, kelewatan projek, mengurangkan produktiviti pembangun, dan meningkatkan tekanan mental. Salah satu penyebab utama penghasilan pembaziran menunggu adalah kurangnya perkongsian pengetahuan dalam Tinjauan Kod Moden (MCR). Walaupun, kajian terdahulu menumpukan kepada perkongsian pengetahuan dalam aktiviti kejuruteraan perisian lain, sedikit bukti kajian terdapat dalam konteks MCR, menyebabkan kurangnya panduan perkongsian pengetahuan dalam MCR bagi membantu jurutera perisian untuk mengurangkan pembaziran menunggu kejuruteraan perisian. Kajian ini telah membangunkan model perkongsian pengetahuan tinjauan kod moden bagi mengurangkan pembaziran menunggu kejuruteraan perisian. Untuk membangunkan model tersebut, faktor perkongsian pengetahuan dalam MCR serta faktor perkongsian pengetahuan paling utama untuk aktiviti MCR telah dikenal pasti. Tinjauan literatur sistematik dijalankan bagi mengenal pasti faktor perkongsian pengetahuan, sub-faktor serta kategori dalam MCR. Garis panduan perkongsian pengetahuan elektronik juga dibangunkan berdasarkan model perkongsian pengetahuan MCR. Empat pakar kejuruteraan perisian mengesahkan senarai faktor perkongsian pengetahuan, sub-faktor dan kategori dalam MCR yang dikenal pasti untuk penyelarasan penamaan, pengelompokan dan sub-kumpulan mereka. Tinjauan Delphi yang melibatkan sepuluh pakar dilaksanakan bagi mengenal pasti faktor perkongsian pengetahuan yang paling berpengaruh dalam aktiviti MCR. Hasil kajian Delphi digunakan untuk membangunkan model perkongsian pengetahuan MCR. Hubungan antara kategori dalam model perkongsian pengetahuan MCR - Individu, Pasukan, Keadaan Kemudahan, Artefak, dan Sosial - dikaji menggunakan analisis regresi. Garis panduan elektronik model perkongsian pengetahuan MCR dibangunkan menggunakan ASP.NET dan pelayan SQL berdasarkan model perkongsian pengetahuan MCR yang telah dibina. Eksperimen telah dijalankan dengan sokongan garis panduan elektronik model perkongsian pengetahuan MCR untuk menilai keberkesanan model yang dibangunkan untuk mengurangkan pembaziran menunggu kejuruteraan perisian. Secara keseluruhannya, kajian ini telah membangunkan model perkongsian pengetahuan MCR, yang terdiri daripada senarai faktor perkongsian pengetahuan yang dinilai dalam MCR, serta faktor perkongsian pengetahuan yang paling berpengaruh dalam aktiviti MCR untuk mengurangkan pembaziran menunggu kejuruteraan perisian.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ACM	-	Association for Computing Machinery
ASP.Net	-	Active Server Pages .Net
CV	-	Coefficient of Variation
CMPPV	-	Composite Mean Perceived Practicality Values
CMPIV	-	Composite Mean Perceived Influence Values
IEEE	-	Institute of Electrical and Electronics Engineers
KSF	-	Knowledge Sharing Factors
KSSbF	-	Knowledge Sharing Sub-factors
KSFP1	-	Knowledge Sharing Factor Paper1
KSFP2	-	Knowledge Sharing Factor Paper2
MCRKSM	-	Modern Code Review Knowledge Sharing Model
MCR	-	Modern Code Review
MPIV	-	Mean Perceived Influence Values
MPPV	-	Mean Perceived Practicality Values
RSN	-	Reviewer Selection and Notification
SLR	-	Systematic Literature Review
SCP	-	Source Code Preparation
SCS	-	Source Code Submission
SCR	-	Source Code Review
SCA	-	Source Code Approval
SQL	-	Structured Query Language
SWEBOK	-	Software Engineering Body of Knowledge
UTM	-	Universiti Teknologi Malaysia
WWW	-	World Wide Web

LIST OF SYMBOLS

σ	-	Standard Deviation
μ	-	Mean
β	-	Coefficient
▲	-	Category
➔	-	Group Cluster under Category
_	-	Data Units Clustered Under a Group
1, 2, 3….	-	Research Statement Number
1a, 1b, 1c…	-	Data Unit in Statement 1
R1	-	Delphi Round 1
R2	-	Delphi Round 2

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Introduction

This chapter provides the details regarding the research background, problem statement, research questions, research objectives, scope of the study, significance, and contributions of the study.

1.2 Background of Research

Software engineering is a cost-effective development of high-quality software within specified resources (Sedano and Ralph, 2017). The success factor of software depends on whether the software solution can fulfil the expectations of the users (Alvertis *et al.*, 2016). Software engineering is a multifaceted socio-technical process that encompasses managing activities for instance software requirement gathering, software modelling and construction, software inspections, and modern code review (Alahyari, Gorschek and Berntsson, 2019), (Sedano and Ralph, 2017). These activities provide ample opportunities to generate software engineering wastes (Alahyari, Gorschek and Berntsson, 2019), (Sedano, 2019), (Sedano and Ralph, 2017).

Software engineering waste refers to an action that does not yield any value to the user (Sedano, 2019). It can also be defined as *"anything that doesn't make it to the release"* (Alahyari, Gorschek and Berntsson, 2019). It can also be demarcated as an activity that utilizes resources but does not deliver quality software and thus not be able to gain the client or end-user satisfaction (Sedano, 2019), (Alahyari, Gorschek and Berntsson, 2019), (Sedano and Ralph, 2017). The wastes which can be generated as a result of software engineering activities can be waiting, needless composite solutions, defect, developing an extra or erroneous features, and mental distress

(Sedano, 2019), (Alahyari, Gorschek and Berntsson, 2019), (Rohan, et al., 2019), (Sedano and Ralph, 2017). It is argued that these wastes should be considered and reduce in every phase of the software development life cycle (Rohan, et al., 2019). It is also conveyed that waiting is one of the major and critical wastes (Alahyari, Gorschek and Berntsson, 2019), (Vlachos, Siachou and Langwallner, 2019). It is reported that *"one of the biggest wastes in software development is usually waiting for the things to happen"* (Poppendieck and Poppendieck, 2003). It is also claimed that if the organization were to consider one waste, they should consider waiting (Alahyari, Gorschek and Berntsson, 2019).

Several causes of waiting waste are reported in the literature. For instance delay in formal approvals (Alahyari, Gorschek and Berntsson, 2019), lack of knowledge sharing (MacLeod *et al.*, 2018), (Sadowski *et al.*, 2018), (Medidi, 2015) poor or unreliable code review and testing, poor code quality, context switching, asynchronous communication (Sedano and Ralph, 2017), large artifact size (Sadowski *et al.*, 2018), (MacLeod *et al.*, 2018), (Thongtanunam *et al.*, 2017), high workload and unavailability of senior developers (Ruangwan *et al.*, 2018), (Bosu *et al.*, 2017), (Kononenko, Baysal and Godfrey, 2016), lack of experience of developers (Ram *et al.*, 2018), (Bosu *et al.*, 2017), (Bosu and Carver, 2014), interactional unfairness (German, Rey and Carlos, 2018), geographical and organizational distance, and lack of tool and process support etc., (MacLeod *et al.*, 2018), (Sadowski *et al.*, 2018), (Medidi, 2015).

Researchers argued that software engineering waiting waste leads to a decrease in developers' productivity, creativity, efficiency, and confidence (Alahyari, Gorschek and Berntsson, 2019), (Sedano, 2019), (German, Rey and Carlos, 2018), (Sedano and Ralph, 2017), (dos Santos and Nunes, 2017). It is conveyed that waiting waste creates a blocking state for other tasks and leads to project delays (Alahyari, Gorschek and Berntsson, 2019), (Ikonen *et al.*, 2010). It is also conveyed that waiting waste increase development cost and effort as well as affect the software quality (Alahyari, Gorschek and Berntsson, 2019), (Sedano and Ralph, 2017), (Menzies *et al.*, 2017), (Behutiye *et al.*, 2017), (Nguyen and Zeng, 2017), (Sarkar and Parnin, 2017).

It is conveyed in the literature that to reduce software engineering waiting waste, it is required to have effective knowledge sharing while performing software engineering activities such as software requirement gathering, software modelling and construction, software inspections, and modern code review (Rohana et al., 2019) (Alahyari, Gorschek and Berntsson, 2019), (Sedano, 2019), (Vlachos, Siachou and Langwallner, 2019), (Sadowski *et al.*, 2018), (MacLeod *et al.*, 2018), (Sedano and Ralph, 2017), (Sambhanthan and Potdar, 2016).

It is also noted that knowledge sharing is dependent on a massive number of factors. These factors arise from different aspects such as people, process and technology and thus these factors need to be explored for effective knowledge sharing and to reduce software engineering waiting waste (Alahyari, Gorschek and Berntsson, 2019), (Sedano, 2019), (Vlachos, Siachou and Langwallner, 2019), (Sadowski *et al.*, 2018), (MacLeod *et al.*, 2018), (Ali and Dominic, 2017), (Sedano and Ralph, 2017), (Sambhanthan and Potdar, 2016), (Medidi, 2015), (Mujtaba, Feldt and Petersen, 2010).

Even Though appreciated work has been performed in the context of knowledge sharing in software engineering (Khalil and Khalil, 2019), (Hsseinoiun *et al.*, 2018), (Anwar *et al.*, 2017), however less attention has been dedicated to the detailed exploration of knowledge sharing factors in the context of modern code review (Sadowski *et al.*, 2018), (MacLeod *et al.*, 2018), (Bosu *et al.*, 2017). Modern code review (MCR) is a significant software engineering activity and a potential means to identify defects, identifying alternative solutions, and improve code quality (MacLeod *et al.*, 2018), (Sadowski *et al.*, 2018), (dos Santos & Nunes, 2017), (Bosu *et al.*, 2017), (Kalyan *et al.*, 2017). It is slightly investigated by researchers concerning factors affecting knowledge sharing, no knowledge sharing model is available for MCR that can support to reduce software engineering waiting waste (Sadowski *et al.*, 2018), (MacLeod *et al.*, 2018), (Bosu *et al.*, 2017), (Bosu, Greiler and Bird, 2015). Therefore, there is a need for a comprehensive model containing knowledge sharing factors affecting knowledge sharing in MCR to reduce software engineering waiting waste.

This demands a modern code review knowledge sharing model to reduce software engineering waiting waste (Alahyari, Gorschek and Berntsson, 2019), (Sedano, 2019), (Sadowski *et al.*, 2018), (Sedano and Ralph, 2017), (MacLeod *et al.*, 2018). Therefore, to reduce software engineering waiting waste, the study aims to develop a modern code review knowledge sharing model by providing a comprehensive list of knowledge sharing factors affecting knowledge sharing in MCR.

1.3 Research Motivation

Waste reduction in software engineering is a complicated task (Rohana et al., 2019). Several wastes produced during software engineering activities such as software requirement gathering, software modelling and construction, software inspections, and modern code review (Rohana et al., 2019) (Alahyari, Gorschek and Berntsson, 2019), (Sedano, 2019), (Sadowski *et al.*, 2018), (MacLeod *et al.*, 2018), (Sedano and Ralph, 2017), (Sambhanthan and Potdar, 2016). It is argued that these wastes should be managed and reduced for all software engineering activities (Rohana et al., 2019). It is also conveyed that to reduce software engineering waiting waste, current research has recommended to focus on knowledge sharing by identifying factors affecting knowledge sharing for software engineering activities, specifically, modern code review (Alahyari, Gorschek and Berntsson, 2019), (Sedano, 2019), (Vlachos, Siachou and Langwallner, 2019), (Sadowski *et al.*, 2018), (MacLeod *et al.*, 2018), (Sedano and Ralph, 2017), (Sambhanthan and Potdar, 2016).

Though valued work has been performed in the context of knowledge sharing concerning software engineering (Khalil and Khalil, 2019), (Hsseinoiun *et al.*, 2018), (Anwar *et al.*, 2017), (Ghobadi, 2015), however, limited attention has been devoted on the thorough exploration of knowledge sharing factors in the context of modern code review (Sadowski *et al.*, 2018), (MacLeod *et al.*, 2018), (Bosu *et al.*, 2017), (Bosu, Greiler and Bird, 2015). The generation of waiting waste creates, a blocking state for other related tasks, delays in project delivery, a decrease in the developers' productivity and increases mental distress (Alahyari, Gorschek and Berntsson, 2019), (Sedano, 2019), (German, Rey and Carlos, 2018). This demands a modern code review

knowledge sharing model to reduce software engineering waiting waste (Alahyari, Gorschek and Berntsson, 2019), (Sedano, 2019), (Sadowski *et al.*, 2018), (MacLeod *et al.*, 2018), (Sedano and Ralph, 2017), (Sambhanthan and Potdar, 2016), (Medidi, 2015), (Mujtaba, Feldt and Petersen, 2010). Hence lack of such research motivated us to develop a modern code review knowledge sharing model to reduce software engineering waiting waste.

1.4 Problem Statement

Software Engineering activities such as software requirement gathering, software modelling and construction, software inspections, and modern code review delivers abundant prospects of generating waiting waste (Rohana et al., 2019), (Alahyari, Gorschek and Berntsson, 2019). As a consequence, it creates, a blocking state for other tasks, project delays, a decrease in the developers' productivity and increases mental distress (Alahyari, Gorschek and Berntsson, 2019), (Sedano, 2019). To reduce software engineering waiting waste, recent research has suggested to have effective knowledge sharing by identifying factors that affect knowledge sharing in software engineering activities, particularly modern code review (Alahyari, Gorschek and Berntsson, 2019), (Sedano, 2019), (Vlachos, Siachou and Langwallner, 2019), (MacLeod *et al.*, 2018). However, the current research in modern code review has been explored to a lesser extent concerning factors influencing knowledge sharing. No knowledge sharing model is available for MCR to reduce software engineering waiting waste.

Therefore, to reduce software engineering waiting waste there is a need to have a knowledge sharing model comprising of knowledge sharing factors for the MCR process. Thus, we are proposing a modern code review knowledge sharing model to reduce software engineering waiting waste. The summarized overview of the problem statement is given in Figure 1.1.

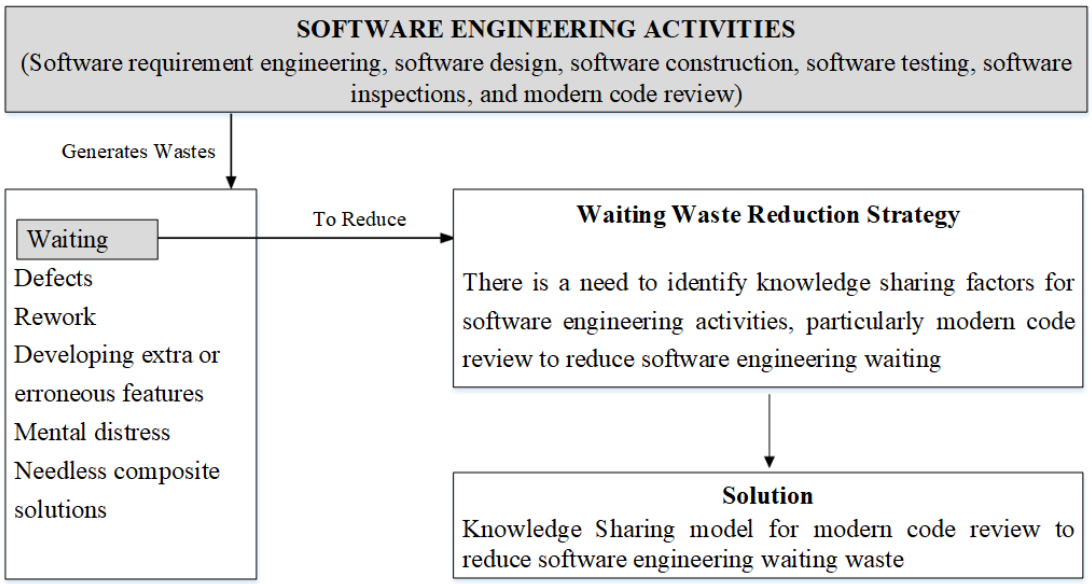

Figure 1.1 Problem statement flow diagram

1.5 Research Questions

This study comprises three research questions.

(a) What knowledge sharing factors of MCR team should be aware of in reducing the software engineering waiting waste?

(b) How the identified knowledge sharing factors can be made effective to the MCR team to reduce software engineering waiting waste?

(c) How modern code review knowledge sharing model can help the MCR team to reduce software engineering waiting waste?

1.6 Objectives of the Study

The study comprises five research objectives. Table 1.1 summarizes the research questions along with the objectives.

(a) To identify the knowledge sharing factors which can help the MCR team to reduce the software engineering waiting waste.

(b) To evaluate the identified list of knowledge sharing factors that can help the MCR team to reduce the software engineering waiting waste.

(c) To develop the modern code review knowledge sharing model to reduce software engineering waiting waste.

(d) To develop an electronic reference guideline of modern code review knowledge sharing model.

(e) To evaluate the effectiveness of the developed modern code review knowledge sharing model to reduce software engineering waiting waste (waiting time).

Table 1.1 Research questions and research objectives

Research Questions	Research Objectives
What knowledge sharing factors of MCR team should be aware of in reducing software engineering waiting waste?	To identify the knowledge sharing factors which can help the MCR team to reduce the software engineering waiting waste.
	To evaluate the identified list of knowledge sharing factors that can help the MCR team to reduce the software engineering waiting waste.
How the identified knowledge sharing factors can be made effective to the MCR team to reduce software engineering waiting waste?	To develop the modern code review knowledge sharing model to reduce software engineering waiting waste.
How modern code review knowledge sharing model can help the MCR team to reduce software engineering waiting waste?	To develop an electronic reference guide of modern code review knowledge sharing model.
	To evaluate the effectiveness of the developed modern code review knowledge sharing model to reduce software engineering waiting waste (waiting time).

1.7 Research Scope

The scope of the study includes the identification of the unique list of knowledge sharing factors in modern code review to reduce software engineering waiting waste. Systematic Literature Review (SLR) was performed following the guidelines given by (Kitchenham and Charters, 2007). Data coding techniques of grounded theory with constant comparison and memoing (Stol, Ralph and Fitzgerald, 2016), (Kathy Charmaz, 2007) were used to generate the unique list of knowledge sharing factors. The considered duration of research papers for SLR was 2013 to 2019.

The expert review was performed to evaluate the list of knowledge sharing factors, sub-factors, and their categories for their naming conventions, grouping, subgrouping, terminologies, and new recommendations. The guidelines given by (Ayyub, 2001) and (Boring *et al.*, 2005) were utilized for expert review. The software engineering professional expert either from industry or academia having experience of 10 or more than 10 years were considered for expert review. Four experts having knowledge of MCR, software engineering wastes, and knowledge sharing were considered for expert review.

The Delphi survey was performed to further evaluate the list of knowledge sharing factors obtained as a result of expert review with industry practices for their grouping, sub-grouping, and naming conventions. The experts were requested to check the practicality of the recognized knowledge sharing factors as well as to identify the most influential knowledge sharing factors concerning MCR activities from industry perspectives. The experts were also requested to suggest new industry-based knowledge sharing factors for MCR. The relationships between the knowledge sharing factors, sub-factors in terms of categories were identified through regression analysis. The guideline given by (Eye and Schuster, 1998) were followed for the regression analysis. A Modern code review knowledge sharing model to reduce software engineering waiting waste was developed after the analysis of Delphi results. Guidelines specified by Murry and Hammons were utilized for conducting the Delphi method (Skulmoski, Hartman and Jennifer Krahn, 2007) and (Hasson, Keeney and

McKenna, 2000). Ten experts with industry experience of more than eight years contributed to the Delphi survey.

The developed modern code review knowledge sharing model to reduce software engineering waiting waste was evaluated for the effectiveness in reducing software engineering waiting waste with the help of an experiment. For the conduction of the experiment, the electronic reference guideline of the modern code review knowledge sharing model was developed. ASP.Net for the development of user interface and Microsoft SQL Server for database development were used. The experiment was conducted with the 28 part-time postgraduate students having industry experience. As it is conveyed that there is no significant difference in the performance of students compared to practitioners (Host, Regnell and Wohlin, 2000). The experiment was conducted in two sessions. The 28 students were divided into two groups each group containing 14 students. In the first session of the experiment, "Group I" and "Group II" performed MCR activities without using the modern code review knowledge sharing model. Later, in the second session the "Group II" was provided with the modern code review knowledge sharing model supported with the electronic reference guideline whereas "Group I" was not provided with the modern code review knowledge sharing model.

1.8 Contributions and Significance of Study

The study contributes to the advancement in the Software engineering body of knowledge (SWEBOK) (Bourque and Fairley, 2014), software engineering waste, and particularly in modern code review. The study contributions are given below.

(a) The first contribution of the study was related to the identification and reporting of knowledge sharing factors for MCR to reduce software engineering waiting waste. Advances to the existing body of knowledge are made possible by performing the SLR with the accessibility of published literature, expert review, and the Delphi survey. As a result, a list of 22 knowledge sharing factors, 135 sub-factors, and 5 categories was recognized.

(b) The second study contribution was connected with the development of the modern code review knowledge sharing model to reduce software engineering waiting waste. As it would specify precisely what knowledge sharing factors would influence software knowledge sharing among MCR team in which specific MCR activity.

(c) The third contribution of the study was the development of the electronic reference guide of the modern code review knowledge sharing model. The electronic guide can support the MCR team in using the modern code review knowledge sharing model and reduce software engineering waiting waste.

1.9 Thesis Outline

The research thesis contains seven chapters. Figure 1.2 provides the outlines of the chapters with a brief explanation.

1.10 Chapter Summary

This chapter provides details concerning the research background, research motivation, and problem statement. It also covers the research questions and research objectives. The research scope, significance and contributions of the study, and thesis outline are presented in the last sections of this chapter.

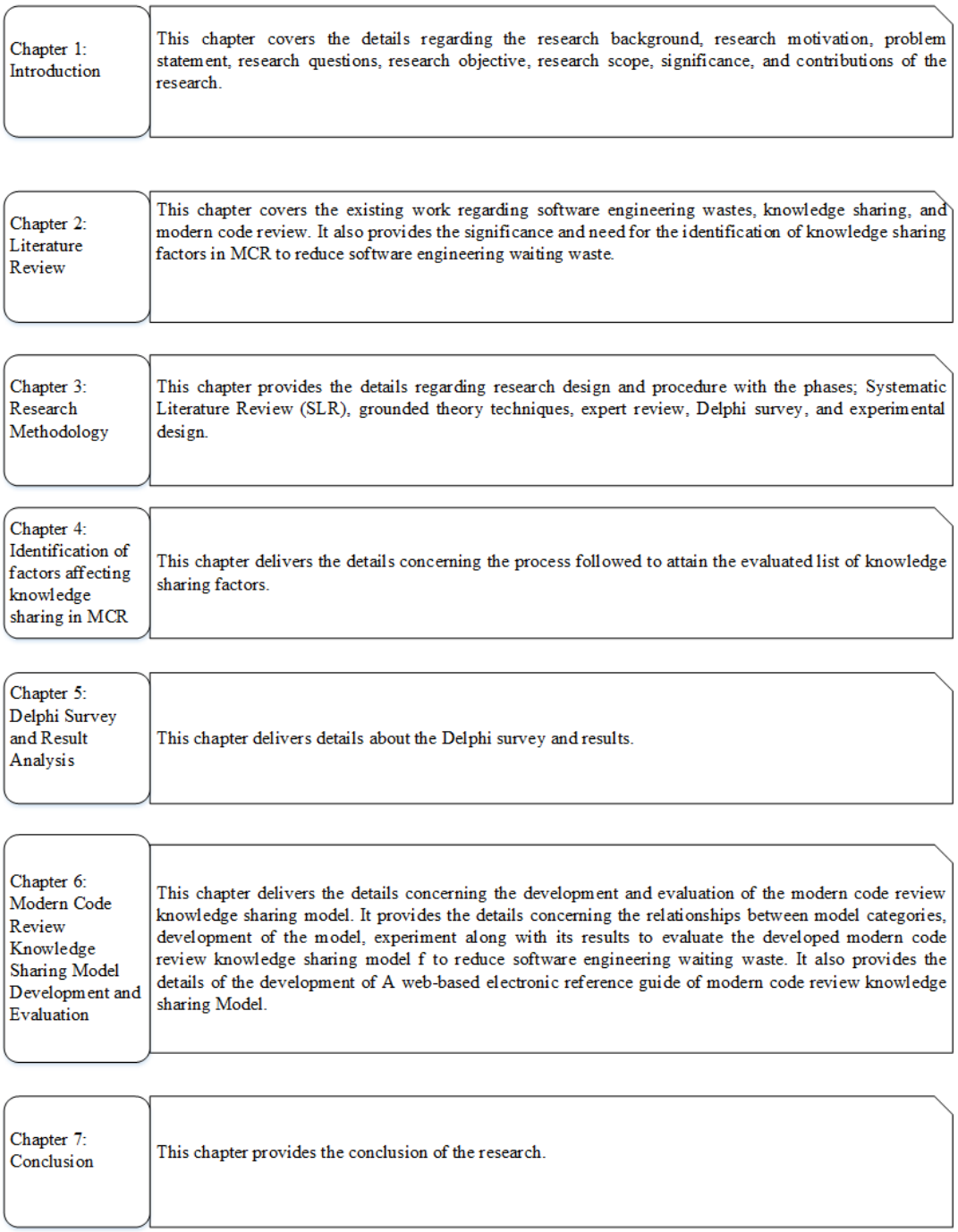

Figure 1.2 Thesis outline

REFERENCES

Adler, M. and Erio, Z. (1996) *Gazing into the Oracle: The Delphi method and its application to social policy and public health*. Jessica Kingsley.

Alahyari, H., Gorschek, T. and Berntsson, R. (2019) 'An exploratory study of waste in software development organizations using agile or lean approaches: A multiple case study at 14 organizations', *Information and Software Technology*. Elsevier, 105(7), pp. 78–94. doi: 10.1016/j.infsof.2018.08.006.

Ali, A. A. and Dominic, P. D. D. (2017) 'The relationship between management support and individual motivation for knowledge sharing practice', in *International Conference on Research and Innovation in Information Systems, ICRIIS*. doi: 10.1109/ICRIIS.2017.8002441.

Alshehri, S. A., Rezgui, Y. and Li, H. (2014) 'Delphi-based consensus study into a framework of community resilience to disaster', *Natural Hazards*, 75(3), pp. 2221–2245. doi: 10.1007/s11069-014-1423-x.

Alvertis, I. *et al.* (2016) 'User involvement in software development processes', *Procedia Computer Science*. Elsevier B.V., 97, pp. 73–83. doi: 10.1016/j.procs.2016.08.282.

Alves, N. S. R. *et al.* (2016) 'Identification and management of technical debt: A systematic mapping study', *Information and Software Technology*, 70, pp. 100–121. doi: 10.1016/j.infsof.2015.10.008.

Anwar, R. *et al.* (2017) 'Conceptual framework for implementation of knowledge sharing in global software development organizations', in *ISCAIE 2017 - 2017 IEEE Symposium on Computer Applications and Industrial Electronics*, pp. 174–178. doi: 10.1109/ISCAIE.2017.8074972.

Armstrong, F., Khomh, F. and Adams, B. (2017) 'Broadcast vs. unicast review technology: Does it matter?', in *in Proc. 10th IEEE International Conference on Software Testing, Verification and Validation*. IEEE, pp. 219–229. doi: 10.1109/ICST.2017.27.

Aurum, A., Petersson, H. and Wohlin, C. (2002) 'State-of-the-art: Software inspections after 25 years', *Software Testing Verification and Reliability*, 12(3), pp. 133–154. doi: 10.1002/stvr.243.

Ayyub, B. (2001) *A practical guide on conducting expert-opinion elicitation of probabilities and consequences for corps facilities*, *Institute for Water Resources, Alexandria, VA, USA*.

B.S. Everitt, A. S. (2011) *The Cambridge dictionary of Statistics*. 4th edn.

Bacchelli, A. and Bird, C. (2013) 'Expectations, outcomes, and challenges of modern code review', in *Proc. International Conference on Software Engineering*. IEEE, pp. 712–721. doi: 10.1109/ICSE.2013.6606617.

Balachandran, V. (2013) 'Reducing human effort and improving quality in peer code reviews using automatic static analysis and reviewer recommendation', *Proceedings - International Conference on Software Engineering*, pp. 931–940. doi: 10.1109/ICSE.2013.6606642.

Baum, T. *et al.* (2016) 'Factors influencing code review processes in industry', *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering - FSE 2016*, pp. 85–96. doi: 10.1145/2950290.2950323.

Behutiye, W. N. *et al.* (2017) 'Analyzing the concept of technical debt in the context of agile software development: A systematic literature review', *Information and Software Technology*, 82, pp. 139–158. doi: 10.1016/j.infsof.2016.10.004.

Di Biase, M., Bruntink, M. and Bacchelli, A. (2016) 'A security perspective on code review: The case of chromium', in *Proc. IEEE 16th International Working Conference on Source Code Analysis and Manipulation*, pp. 21–30. doi: 10.1109/SCAM.2016.30.

Bird, C., Carnahan, T. and Greiler, M. (2015) 'Lessons learned from building and deploying a code review analytics platform', in *Proc. IEEE International Working Conference on Mining Software Repositories*, pp. 191–201. doi: 10.1109/MSR.2015.25.

Boring, R. *et al.* (2005) *Simplified expert elicitation guideline for risk assessment of operating events*, *National Laboratory INL*. Available at: http://www.inl.gov/technicalpublications/documents/3310952.pdf.

Bosu, A. (2013) 'Modeling modern code review practices in open source software development organizations', in *Proc. IDoESE '13 Baltimore*. Maryland.

Bosu, A. *et al.* (2014) 'Peer impressions in open source organizations: A survey', *Journal of Systems and Software*. Elsevier Inc., 94, pp. 4–15. doi: 10.1016/j.jss.2014.03.061.

Bosu, A. *et al.* (2017) 'Process aspects and social dynamics of contemporary code review: Insights from open source development and industrial practice at Microsoft', *IEEE Transactions on Software Engineering*, 43(1), pp. 56–75. doi: 10.1109/TSE.2016.2576451.

Bosu, A. and Carver, J. C. (2013) 'Impact of peer code review on peer impression formation: A survey', *International Symposium on Empirical Software Engineering and Measurement*, pp. 133–142. doi: 10.1109/ESEM.2013.23.

Bosu, A. and Carver, J. C. (2014) 'Impact of developer reputation on code review outcomes in OSS projects', *Proc. 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pp. 1–10. doi: 10.1145/2652524.2652544.

Bosu, A., Greiler, M. and Bird, C. (2015) 'Characteristics of useful code reviews: An empirical study at Microsoft', in *Proc. IEEE International Working Conference on Mining Software Repositories*, pp. 146–156. doi: 10.1109/MSR.2015.21.

Bourque, P. and Fairley, R. E. (2014) *Guide to the software engineering - Body of knowledge.*, *IEEE Computer Society*. doi: 10.1234/12345678.

Bryant, A. and Charmaz, K. (2010) *The SAGE Handbook of Grounded Theory*. SAGE Publication Limited.

Buse, R. P. L. and Weimer, W. R. (2010) 'Learning a metric for code readability', *IEEE Transactions on Software Engineering*, 36(4), pp. 546–558. doi: 10.1109/TSE.2009.70.

Carroll, J. M. *et al.* (2006) 'Awareness and teamwork in computer-supported collaborations', *Interacting with Computers*, 18(1 SPEC. ISS.), pp. 21–46. doi: 10.1016/j.intcom.2005.05.005.

Chaves, M. S., Scornavacca, E. and Fowler, D. (2018) 'Affordances of social media in knowledge sharing in intra-organizational information technology projects', *Lecture Notes in Information Systems and Organisation*, 23, pp. 35–47. doi: 10.1007/978-3-319-62051-0_4.

Chen, X. *et al.* (2017) 'Managing knowledge sharing in distributed innovation from the perspective of developers: empirical study of open source software projects in China', *Technology Analysis and Strategic Management*, 29(1), pp. 1–22. doi: 10.1080/09537325.2016.1194387.

Clarke, P. *et al.* (2016) 'A complexity theory viewpoint on the software development process and situational context', in *IEEE/ACM International Conference on*

Software and System Processes. IEEE/ACM. doi: 10.1145/2904354.2904369.

Cohen, J. (1988) *Statistical power analysis for the behavioral science*. 2nd edn. New: Lawrence Erlbaum Associates.

Contreras-Pacheco, O. E., Claasen, C. and Nishant, R. (2017) 'Knowledge sharing among engineers: An empirical examination', in *2017 IEEE Technology and Engineering Management Society Conference, TEMSCON 2017*, pp. 260–266. doi: 10.1109/TEMSCON.2017.7998386.

Cooke, N. J. *et al.* (2000) 'Measuring team knowledge', *Human Factors*, 42(1), pp. 151–173. doi: 10.1518/001872000779656561.

Corral, L. *et al.* (2014) 'Code review analytics: WebKit as case study', in *IFIP Advances in Information and Communication Technology*, pp. V–VI. doi: 10.1007/978-3-642-55128-4.

Czerwonka, J. (2018) 'CodeFlow: Improving the code review process at Microsoft', *Queue*, 16(5), pp. 1–20. doi: 10.1145/3291276.3292420.

Czerwonka, J., Greiler, M. and Tilford, J. (2015) 'Code Reviews Do Not Find Bugs. How the Current Code Review Best Practice Slows Us Down', in *Proc. International Conference on Software Engineering*, pp. 27–28. doi: 10.1109/ICSE.2015.131.

Dajani, J. S., Sincoff, M. Z. and Talley, W. K. (1979) 'Stability and agreement criteria for the termination of Delphi studies', *Technological Forecasting and Social Change*, 13(1), pp. 83–90. doi: 10.1016/0040-1625(79)90007-6.

Daud, M. A. (2000) *The students' perceptions on the factors that motivate them to participate in accounting class*.

Delbecq, A., Ven, A. and Gustafson, D. (1986) *Group techniques for program planning: A guide to nominal group and Delphi processes*. 1st edn. Green Briar Press.

Van Deursen, A. J. A. M., Verlage, C. and Van Laar, E. (2019) 'Social network site skills for communication professionals: Conceptualization, operationalization, and an empirical investigation', *IEEE Transactions on Professional Communication*, 62(1), pp. 43–54. doi: 10.1109/TPC.2018.2867168.

Dheyaa, A., Salah, H. and Ali, N. (2014) 'Factors influencing knowledge sharing in organizations: A literature review', *International Journal of Science and Research*, 3(9), pp. 1314–1319. Available at: www.ijsr.net.

Ducheneaut, N. (2005) 'Socialization in an open source software community: A socio-

technical analysis', *Computer Supported Cooperative Work: CSCW: An International Journal*, 14(4), pp. 323–368. doi: 10.1007/s10606-005-9000-1.

Ebert, F. *et al.* (2017) 'Confusion detection in code reviews', in *Proc. IEEE International Conference on Software Maintenance and Evolution*, pp. 549–553. doi: 10.1109/ICSME.2017.40.

Ebert, F. *et al.* (2018) 'Communicative intention in code review questions'. doi: 10.1136/jfprhc-2012-100335.

Ebert, F. *et al.* (2019) 'Confusion in code reviews: reasons, impacts, and coping strategies', in *SANER- Proceedings of the 2019 IEEE 26th International Conference on Software Analysis, Evolution, and Reengineering*. IEEE, pp. 49–60. doi: 10.1109/SANER.2019.8668024.

Efstathiou, V. and Spinellis, D. (2018) 'Code review comments: language matters', in *Proceedings of the 40th International Conference on Software Engineering*. ACM, pp. 69–72.

English, J. M. and Kernan, G. L. (1976) 'The prediction of air travel and aircraft technology to the year 2000 using the Delphi method', *Transportation Research*, 10(1), pp. 1–8. doi: 10.1016/0041-1647(76)90094-0.

Eye, A. and Schuster, C. (1998) *Regression Analysis for Social Sciences*.

Faegri, T. E., Stray, V. and Moe, N. B. (2016) 'Shared knowledge in virtual software teams: A preliminary framework', *Proceedings - 11th IEEE International Conference on Global Software Engineering, ICGSE 2016*, pp. 174–178. doi: 10.1109/ICGSE.2016.22.

Fagan, M. E. (1999) 'Design and code inspections to reduce errors in program development', *IBM Systems Journal*, 38(2.3), pp. 258–287. doi: 10.1147/sj.382.0258.

Fakhoury, S. (2018) 'The effect of poor source code lexicon and readability on developers' cognitive load', in *Proc. ICPC*, pp. 286–296. doi: 10.1145/3196321.3196347.

Fejzer, M., Przymus, P. and Stencel, K. (2018) 'Profile based recommendation of code reviewers', *Journal of Intelligent Information Systems*. Journal of Intelligent Information Systems, 50(3), pp. 597–619. doi: 10.1007/s10844-017-0484-1.

Fountaine, A. and Sharif, B. (2017) 'Emotional awareness in software development: Theory and measurement', *Proceedings - 2017 IEEE/ACM 2nd International Workshop on Emotion Awareness in Software Engineering, SEmotion 2017*, pp.

28–31. doi: 10.1109/SEmotion.2017.12.

Fracz, W. and Jacek, D. (2016) 'Experimental Validation of Source Code Reviews on Mobile Devices', *CEUR Workshop Proceedings*, 1603(July), pp. 1–8. doi: 10.1007/978-3-319-62404-4.

Gachechiladze, D. *et al.* (2017) 'Anger and its direction in collaborative software development', *Proc.- 2017 IEEE/ACM 39th International Conference on Software Engineering: New Ideas and Emerging Results Track*, pp. 11–14. doi: 10.1109/ICSE-NIER.2017.18.

Ge, X. *et al.* (2017) 'Refactoring-Aware code review', in *Proceedings of IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC*, pp. 71–79. doi: 10.1109/VLHCC.2017.8103453.

German, D. M., Rey, U. and Carlos, J. (2018) '" 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*, pp. 523–534.

Ghobadi, S. (2015) 'What drives knowledge sharing in software development teams: A literature review and classification framework', *Information and Management*. Elsevier B.V., 52(1), pp. 82–97. doi: 10.1016/j.im.2014.10.008.

Glasser, B. G. (1992) *Basics of Grounded Theory Analysis: Emergence Vs. Forcing*. Sociology Press.

Glasser, B. G. (1998) *Doing grounded theory: Issues & discussion*. Sociology Press.

Goldman, K. *et al.* (2008) 'Identifying important and difficult concepts in introductory computing courses using a delphi process', *SIGCSE'08 - Proceedings of the 39th ACM Technical Symposium on Computer Science Education*, pp. 256–260. doi: 10.1145/1352135.1352226.

Gousios, G., Storey, M.-A. and Bacchelli, A. (2016) 'Work practices and challenges in pull-based development', in *Proc. 38th International Conference on Software Engineering*, pp. 285–296. doi: 10.1145/2884781.2884826.

Gracht, H. (2012) 'Consensus measurement in Delphi studies. Review and implications for future quality assurance', *Technological Forecasting and Social Change*. Elsevier Inc., 79(8), pp. 1525–1536. doi: 10.1016/j.techfore.2012.04.013.

Greatorex, Jane & Dexter, T. (2000) 'An accessible analytic approach for investigating what happens between the rounds of a Delphi study', *Journal of Advanced*

Nursing, 32, pp. 1016–24. doi: 1016-24. 10.1046/j.1365-2648.2000.01569.x.

Grisham, T. (2009) 'The Delphi technique: a method for testing complex and multifaceted topics', *International Journal of Managing Projects in Business*, 2(1), pp. 112–130. doi: 10.1108/17538370910930545.

Guo, B. and Song, M. (2017) '"Interactively decomposing composite changes to support code review and regression testing"', in *Proceedings - International Computer Software and Applications Conference*, pp. 118–127. doi: 10.1109/COMPSAC.2017.153.

Harold Sackman (1974) *Delphi critique; expert opinion, forecasting, and group process*. Lexington Books.

Hasson, F., Keeney, S. and McKenna, H. (2000) 'Research guidelines for the Delphi survey technique', *Journal of Advanced Nursing*, 32(4), pp. 1008–1015. doi: 10.1046/j.1365-2648.2000.t01-1-01567.x.

Hatcher, T. and Colton, S. (2007) 'Using the internet to improve HRD research: The case of the web-based Delphi research technique to achieve content validity of an HRD-oriented measurement', *Journal of European Industrial Training*, 31(7), pp. 570–587. doi: 10.1108/03090590710820060.

Holey, E. A. *et al.* (2007) 'An exploration of the use of simple statistics to measure consensus and stability in Delphi studies', *BMC Medical Research Methodology*, 7(February). doi: 10.1186/1471-2288-7-52.

Holvitie, J., Leppänen, V. and Hyrynsalmi, S. (2014) 'Technical debt and the effect of agile software development practices on it - An industry practitioner survey', in *Proc. 2014 6th IEEE International Workshop on Managing Technical Debt*, pp. 35–42. doi: 10.1109/MTD.2014.8.

Host, M., Regnell, B. and Wohlin, C. (2000) 'Using students as subjects — A comparative study of students and professionals in lead-time impact assessment', *Empirical Software Engineering*, 5, pp. 201–214.

Hsseinoiun, S. *et al.* (2018) 'Information system success and knowledge grid integration in facilitating knowledge sharing among big data community', in *Proceedings - 2018 4th International Conference on Information Retrieval and Knowledge Management: Diving into Data Sciences, CAMP 2018*. IEEE, pp. 211–215. doi: 10.1109/INFRKM.2018.8464790.

Ikonen, M. *et al.* (2010) 'Exploring the sources of waste in Kanban software development projects', in *Proceedings - 36th EUROMICRO Conference on*

Software Engineering and Advanced Applications, SEAA 2010, pp. 376–381. doi: 10.1109/SEAA.2010.40.

ISO/IEC/IEEE (2013) 'ISO/IEC/IEEE Systems and software engineering -- System life cycle processes', *Iso/Iec/Ieee P15288-Fdis-1412*.

ISO/IEC and IEEE (2010) 'ISO/IEC/IEEE 24765:2010 - Systems and software engineering -- Vocabulary', *Iso/Iec Ieee*. doi: 10.1109/IEEESTD.2010.5733835.

Jiang, Y., Adams, B. and German, D. M. (2013) 'Will my patch make it? And how fast?: Case study on the linux kernel', in *Proc. IEEE International Working Conference on Mining Software Repositories*, pp. 101–110. doi: 10.1109/MSR.2013.6624016.

Juliet, C. and Strauss, A. (2008) *Basics of qualitative research: Techniques and procedures for developing grounded theory.* 3rd edn.

Kalyan, A. *et al.* (2017) 'A collaborative code review platform for GitHub', in *Proc. IEEE International Conference on Engineering of Complex Computer Systems*, pp. 191–196. doi: 10.1109/ICECCS.2016.032.

Kangas, M. *et al.* (2018) 'Why do managers leave their organization? Investigating the role of ethical organizational culture in managerial turnover', *Journal of Business Ethics*, 153(3), pp. 707–723. doi: 10.1007/s10551-016-3363-8.

Kathy Charmaz (2007) *Situational requirement engineering in global software development*.

Khalil, C. and Khalil, S. (2019) 'Exploring knowledge management in agile software development organizations', *International Entrepreneurship and Management Journal*. International Entrepreneurship and Management Journal. doi: 10.1007/s11365-019-00582-9.

Khan, H. H. (2015) *Situational requirement engineering model for global.* University Technology Malaysia.

Kitchenham *et al.* (2002) *Preliminary guidelines for empirical research in software engineering*, *IEEE Transactions on Software Engineering*. doi: 10.1109/TSE.2002.1027796.

Kitchenham, B. and Charters, S. (2007) *Guidelines for performing systematic literature reviews in software engineering*. doi: 10.1145/1134285.1134500.

Knapp, K. and Ferrante, C. (2012) 'Policy Awareness, Enforcement and Maintenance: Critical to Information Security Effectiveness in Organizations', *Journal of Management Policy and Practice*, 13(5), pp. 66–80.

Kollanus, S. and Koskinen, J. (2007) ‘Survey of software inspection’, *Information Systems*, 35, pp. 1991–2005. doi: 10.2174/1874107X00903010015.

Kononenko, O. *et al.* (2015) ‘Investigating code review quality: Do people and participation matter?’, in *Proc. IEEE 31st International Conference on Software Maintenance and Evolution*, pp. 111–120. doi: 10.1109/ICSM.2015.7332457.

Kononenko, O. *et al.* (2018) ‘Studying pull request merges : A case study of shopify ’s active merchant’’, in *Proc. 40th International Conference on Software Engineering: Software Engineering in Practice*, pp. 124–133. doi: 10.1145/3183519.3183542.

Kononenko, O., Baysal, O. and Godfrey, M. W. (2016) ‘Code review quality: How developers see it’, in *Proc. International Conference on Software Engineering*. ACM, pp. 1028–1038. doi: 10.1145/2884781.2884840.

Kovalenko, V. *et al.* (2018) ‘Does reviewer recommendation help developers?’, *IEEE Transactions on Software Engineering*. IEEE, PP(c), p. 1. doi: 10.1109/TSE.2018.2868367.

Lal, H. and Pahwa, G. (2017) ‘Code review analysis of software system using machine learning techniques’, in *in Proc. 11th International Conference on Intelligent Systems and Control*. IEEE, pp. 8–13. doi: 10.1109/ISCO.2017.7855962.

Lanford, H. W. (1972) *Technological forecasting methodologies; a synthesis*. New York: American Management Association.

Lee, A., Carver, J. C. and Bosu, A. (2017) ‘Understanding the impressions, motivations, and barriers of one time code contributors to FLOSS projects: A survey’, in *in Proce. IEEE/ACM 39th International Conference on Software Engineering*, pp. 187–197. doi: 10.1109/ICSE.2017.25.

Lin, B., Robles, G. and Serebrenik, A. (2017) ‘Developer turnover in global, industrial open source projects: Insights from applying survival analysis’, *Proceedings - 2017 IEEE 12th International Conference on Global Software Engineering, ICGSE 2017*, pp. 66–75. doi: 10.1109/ICGSE.2017.11.

Luxton-reilly, A., Lewis, A. and Plimmer, B. (2018) ‘Automating the software inspection process’, in *Proc. 20th Australasian Computing Education Conference*, pp. 45–52. doi: 10.1145/3160489.3160498.

Macdonald, F. *et al.* (1996) ‘Automating the Software Inspection Process’, *Automated Software Engineering*, 3(3–4), pp. 193–218. doi: 10.1007/BF00132566.

MacLeod, L. *et al.* (2017) ‘Code reviewing in the trenches: Challenges and best

practices', *IEEE Software*. doi: 10.1109/MS.2017.265100500.

MacLeod, L. *et al.* (2018) 'Code reviewing in the trenches: Challenges and best practices', *IEEE Software*. IEEE, 35(4), pp. 34–42. doi: 10.1109/MS.2017.265100500.

Magnuson, L. A. (2012) *A Delphi study to understand relational bonds in supervision and their effect on rehabilitation counselor disclosure in the public rehabilitation program*, *ProQuest Dissertations and Theses*. University of Iowa.

Mark, K. (2006) *Designing an effective survey*.

Marlow, J., Dabbish, L. and Herbsleb, J. (2013) 'Impression formation in online peer production : Activity traces and personal profiles in GitHub', in *Proc. 16th ACM Conference on Computer Supported Cooperative Work*, pp. 117–128. doi: 10.1145/2441776.2441792.

Medidi, P. (2015) *Waste in Lean Software Development: a Root Cause Analysis*. Blekinge Institute of Technology, Karlskrona, Sweden.

Meneely, A. *et al.* (2014) 'An empirical investigation of socio-technical code review metrics and security vulnerabilities', *6th International Workshop on Social Software Engineering, SSE 2014 - Proceedings*, pp. 37–44. doi: 10.1145/2661685.2661687.

Menzies, T. *et al.* (2017) 'Are delayed issues harder to resolve? Revisiting cost-to-fix of defects throughout the lifecycle', *Empirical Software Engineering*, 22(4), pp. 1903–1935. doi: 10.1007/s10664-016-9469-x.

Morales, R., McIntosh, S. and Khomh, F. (2015) 'Do code review practices impact design quality? A case study of the Qt, VTK, and ITK projects', in *Proc. IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, pp. 171–180. doi: 10.1109/SANER.2015.7081827.

Morrison, E. W. (2002) 'Newcomers' relationships: The role of social network ties during socialization', *Academy of Management Journal*, 45(6), pp. 1149–1160. doi: 10.2307/3069430.

Mujtaba, S., Feldt, R. and Petersen, K. (2010) 'Waste and lead time reduction in a software product customization process with value stream maps', *Proceedings of the Australian Software Engineering Conference, ASWEC*, pp. 139–148. doi: 10.1109/ASWEC.2010.37.

Naoum, S. G. (2012) *Dissertation research and writing for construction students, Second Edition*. 2nd edn. Oxford : Butterworth-Heinemann. doi:

10.4324/9780080467047.

Narayanam, R. and Narahari, Y. (2009) 'Stability and efficiency of social networks with strategic, resource constrained nodes', *2009 IEEE Conference on Commerce and Enterprise Computing, CEC 2009*, pp. 188–193. doi: 10.1109/CEC.2009.59.

Nazir, S., Fatima, N. and Malik, S. (2008) 'Effective hybrid review process (EHRP)', *Proceedings - International Conference on Computer Science and Software Engineering, CSSE 2008*, 2, pp. 763–771. doi: 10.1109/CSSE.2008.1417.

Nguyen, T. A. and Zeng, Y. (2017) 'Effects of stress and effort on self-rated reports in experimental study of design activities', *Journal of Intelligent Manufacturing*. Springer US, 28(7), pp. 1609–1622. doi: 10.1007/s10845-016-1196-z.

NSPE (2013) 'Professional engineering body of knowledge', p. 61. Available at: https://www.nspe.org/sites/default/files/resources/nspe-body-of-knowledge.pdf.

Ouni, A., Kula, R. G. and Inoue, K. (2017) 'Search-based peer reviewers recommendation in modern code review', in *Proc. - IEEE International Conference on Software Maintenance and Evolution*, pp. 367–377. doi: 10.1109/ICSME.2016.65.

Pessôa, M. V. P., Seering, W. and Rebentisch, E. (2008) 'Understanding the waste net: A method for waste elimination prioritization in product development', *Proceedings of DETC '08*, 55(21), pp. 1–9. doi: 10.1007/978-1-84882-762-2_22.

Peter C. Rigby, Yue Cai Zhu, Samuel M. Donadelli, A. M. (2016) 'Quantifying and mitigating turnover-induced knowledge loss: Case studies of chrome and a project at Avaya', in *Proc. IEEE/ACM 38th International Conference on Software Engineering*, pp. 1006–1016. doi: 10.1145/2884781.2884851.

Pletea, D., Vasilescu, B. and Serebrenik, A. (2014) 'Security and emotion: Sentiment analysis of security discussions on GitHub', *11th Working Conference on Mining Software Repositories, MSR 2014 - Proceedings*, pp. 348–351. doi: 10.1145/2597073.2597117.

Polit, D. F. and Beck, C. T. (2011) *Nursing research: generating and assessing evidence for nursing practice. Lippincott Williams and Wilkins, Philadelphia*. 9th edn. LWW.

Poppendieck, M. and Poppendieck, T. (2003) *Lean software development: An agile*

toolkit, *Addilson-Wesley*. doi: 10.1109/MC.2003.1220585.

Power, K. and Conboy, K. (2014) 'Impediments to flow: Rethinking the lean concept of "waste" in modern software development BT - 15th International Conference on Agile Software Development, XP 2014, May 26, 2014 - May 30, 2014', 179 LNBIP, pp. 203–217. doi: 10.1007/978-3-319-06862-6.

Ram, A. *et al.* (2018) 'What makes a code change easier to review? An empirical investigation on code change reviewability', in *Proc. ESEC/FSE*. ACM. doi: 10.5281/zenodo.1323659.

Rigby *et al.* (2014) 'Peer review on open-source software projects: Parameters, statistical models, and theory', *ACM Transactions on Software Engineering and Methodology*, 23(4). doi: 10.1145/2594458.

Rigby, P. C. (2011) *Understanding open source software peer review: Review processes, parameters and statistical models , and underlying behaviours and mechanisms*. University of Victoria.

Rigby, P. C. and Bird, C. (2013) 'Convergent contemporary software peer review practices categories and subject descriptors', in *Proc. ESEC/FSE*, pp. 202–212. Rigby, P. C., German, D. M. and Storey, M.-A. (2008) 'Open source software peer review practices: a case study of the apache server', *Proceedings of the 30th international conference on Software engineering*, pp. 541–550. doi: 10.1145/1368088.1368162.

Rosai, J. (1978) *A Guide to the Project Management Body of Knowledge*, *American Journal of Clinical Pathology*. doi: 10.1093/ajcp/69.5.475.

Ruangwan, S. *et al.* (2018) 'The impact of human factors on the participation decision of reviewers in modern code review', *Empirical Software Engineering manuscript*, pp. 1–43. Available at: http://arxiv.org/abs/1806.10277.

Sadowski, C. *et al.* (2018) 'Modern code review: : A case study at google', in *Proc. ACM/IEEE 40th International Conference on Software Engineering: Software Engineering in Practice*, pp. 181–190. doi: 10.1145/3183519.3183525.

Saide, Indrajit, R. E. and Hafiza, W. (2017) 'Information technology and individual factors on knowledge sharing activities', *2017 2nd International Conference on Knowledge Engineering and Applications (ICKEA)*, pp. 162–165. doi: 10.1109/ICKEA.2017.8169922.

Salam, M. and Khan, S. U. (2017) 'Risks mitigation practices for multi-sourcing vendors in green software development', in *Proc. of the Pakistan Academy of*

Sciences, pp. 71–87.

Sambhanthan, A. and Potdar, V. (2016) 'Waste management strategies for Software Development companies: An explorative text analysis of business sustainability reports', *2016 IEEE/ACIS 14th International Conference on Software Engineering Research, Management and Applications, SERA 2016*, pp. 179–184. doi: 10.1109/SERA.2016.7516144.

dos Santos, E. W. and Nunes, I. (2017) 'Investigating the effectiveness of peer code review in distributed software development', in *Proc. 31st Brazilian Symposium on Software Engineering*. ACM, pp. 84–93. doi: 10.1145/3131151.3131161.

Sarkar, S. and Parnin, C. (2017) 'Characterizing and predicting mental fatigue during programming tasks', *Proceedings - 2017 IEEE/ACM 2nd International Workshop on Emotion Awareness in Software Engineering, SEmotion 2017*, pp. 32–37. doi: 10.1109/SEmotion.2017.2.

Sarker, Saonee, Ahuj, M. and Sarker, Suprateek (2018) 'Work-life conflict of globally distributed software development personnel: An empirical investigation using Border Theory', *Information Systems Research*, 29(1), pp. 103–126. doi: 10.1287/isre.2017.0734.

Savu, I., Popa, C. L. and Cotet, C. E. (2017) 'Mitigating friction in multicultural virtual organizations / teams', *Annals of DAAAM and Proceedings of the International DAAAM Symposium*, (January 2017), pp. 737–742. doi: 10.2507/28th.daaam.proceedings.104.

Scott, L. *et al.* (2002) 'Understanding the use of an electronic process guide', *Information and Software Technology*, 44(10), pp. 601–616. doi: 10.1016/S0950-5849(02)00080-0.

Sedano, T. (2019) 'Removing Software Development Waste to Improve Productivity', in *Rethinking Productivity in Software Engineering*. Apress, pp. 221–240. doi: 10.1007/978-1-4842-4221-6.

Sedano, T. and Ralph, P. (2017) 'Software Development Waste', in *Proc. IEEE/ACM 39th International Conference on Software Engineering*. doi: 10.1109/ICSE.2017.20.

Sedano, T., Ralph, P. and Peraire, C. (2016) 'Practice and perception of team code ownership', in *Proc. EASE '16,*.

Shah, H. A. and Kalaian, S. A. (2009) 'Which is the best parametric statistical method for analyzing delphi data?', *Journal of Modern Applied Statistical Methods*,

8(1), pp. 226–232. doi: 10.22237/jmasm/1241137140.

Shariff, N. J. (2015) 'Utilizing the Delphi survey approach: A review', *Journal of Nursing & Care*, 04(03). doi: 10.4172/2167-1168.1000246.

Shimagaki, J. *et al.* (2016) 'A study of the quality-impacting practices of modern code review at Sony mobile', in *Proc. 38th International Conference on Software Engineering Companion*, pp. 212–221. doi: 10.1145/2889160.2889243.

Singh, D. *et al.* (2017) 'Evaluating how static analysis tools can reduce code review effort', *IEEE Symposium on Visual Languages and Human-Centric Computing*, pp. 101–105. doi: 10.1109/VLHCC.2017.8103456.

Skulmoski, G. J., Hartman, F. T. and Jennifer Krahn (2007) 'The Delphi method for graduate research', *Journal of Information Technology Education*, 6. doi: 10.1007/3-540-47847-7_10.

Sommerville, I. (2013) *Software Engineering*, *Clinical Engineering: A Handbook for Clinical and Biomedical Engineers*. doi: 10.1016/B978-0-12-396961-3.00009-3.

Stol, K.-J., Ralph, P. and Fitzgerald, B. (2016) 'Grounded theory in software engineering research', (October 2017), pp. 120–131. doi: 10.1145/2884781.2884833.

Taber, K. S. (2018) 'The Use of Cronbach's Alpha When Developing and Reporting Research Instruments in Science Education', *Research in Science Education*. Research in Science Education, 48(6), pp. 1273–1296. doi: 10.1007/s11165-016-9602-2.

Tao, Y., Han, D. and Kim, S. (2014) 'Writing acceptable patches: An empirical study of open source project patches', *Proceedings - 30th International Conference on Software Maintenance and Evolution, ICSME 2014*, pp. 271–280. doi: 10.1109/ICSME.2014.49.

Thongtanunam, P. *et al.* (2014) 'REDA: A web-based visualization tool for analyzing modern code review dataset', *Proceedings - 30th International Conference on Software Maintenance and Evolution, ICSME 2014*, pp. 605–608. doi: 10.1109/ICSME.2014.106.

Thongtanunam, P. *et al.* (2016) 'Revisiting code ownership and its relationship with software quality in the scope of modern code review', in *Proceedings of the 38th International Conference on Software Engineering - ICSE '16*, pp. 1039–1050. doi: 10.1145/2884781.2884852.

Thongtanunam, P. *et al.* (2017) 'Review participation in modern code review', *Empirical Software Engineering*, 22(2), pp. 768–817. doi: 10.1007/s10664-016-9452-6.

Thung, F. *et al.* (2013) 'Network structure of social coding in GitHub', *Proceedings of the European Conference on Software Maintenance and Reengineering, CSMR*, pp. 323–326. doi: 10.1109/CSMR.2013.41.

Tobias, D. (2017) 'Continuous Code Reviews', in *Proc. Programming'17, Brussels, Belgium*. ACM, pp. 5–7.

Tottossy, A. P. (2005) *Teacher selection: A Delphi study*.

Travassos, G. H. (2014) 'Software defects: Stay away from them. Do inspections!', in *Proc. 2014 9th International Conference on the Quality of Information and Communications Technology*, pp. 1–7. doi: 10.1109/QUATIC.2014.8.

Tsay, J., Dabbish, L. and Herbsleb, J. (2014) 'Influence of Social and Technical Factors for Evaluating Contribution in GitHub', in *Proc. International Conference on Software Engineering*, pp. 356–366. doi: 10.1145/2568225.2568315.

Tymchuk, Y., Mocci, A. and Lanza, M. (2015) 'Code Review : Veni , ViDI , Vici', in *Proc. 171 SANER, Montréal, Canada*, pp. 151–160.

Urrego, J. *et al.* (2014) 'Archinotes: A global agile architecture design approach', *Lecture Notes in Business Information Processing*, 179 LNBIP, pp. 302–311. doi: 10.1007/978-3-319-06862-6.

Vasanthapriyan, S. *et al.* (2017) 'An ontology-based knowledge sharing portal for software testing', *2017 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pp. 472–479. doi: 10.1109/QRS-C.2017.82.

Vlachos, I., Siachou, E. and Langwallner, E. (2019) 'A perspective on knowledge sharing and lean management: an empirical investigation', *Knowledge Management Research and Practice*. Taylor & Francis, 00(00), pp. 1–16. doi: 10.1080/14778238.2019.1589399.

Wen, R. *et al.* (2018) 'BLIMP tracer: Integrating build impact analysis with code review', in *Proc. IEEE International Conference on Software Maintenance and Evolution*, pp. 685–694. doi: 10.1109/ICSME.2018.00078.

Winter, J. C. F. (2013) 'Using the student's t-test with extremely small sample sizes', *Practical Assessment, Research and Evaluation*, 18(10), pp. 1–12.

Wohlin, C. *et al.* (2000) *Experimentation in software engineering*, *Springer*

International Publishing. Edited by V. Basili R. Springer. doi: 10.1007/978-1-4615-4125-7.

Wong, Y. K. (2006) *Modern software review: Techniques and technologies*.

Xia, Z. *et al.* (2017) 'A hybrid approach to code reviewer recommendation with collaborative filtering', in *SoftwareMining 2017, Urbana-Champaign, IL, USA*. IEEE, pp. 24–31.

Yang, X. *et al.* (2016) 'Peer review social network (PeRSoN) in open source projects', *IEICE Transactions on Information and Systems*, E99D(3), pp. 661–670. doi: 10.1587/transinf.2015EDP7261.

Yang, Y. N. (2000) 'Convergence on the guidelines for designing a web- based Art-teacher education curriculum : A Delphi Study', *American Educational Research Association*, (Cv).

Yu, Y. *et al.* (2014) 'Reviewer recommender of pull-requests in GitHub', in *Proceedings - 30th International Conference on Software Maintenance and Evolution, ICSME 2014*, pp. 609–612. doi: 10.1109/ICSME.2014.107.

Yu, Y. *et al.* (2015) 'Reviewer recommendation for pull-requests in GitHub: What can we learn from code review and bug assignment?', *Information and Software Technology*. Elsevier B.V., 000, pp. 1–15. doi: 10.1016/j.infsof.2016.01.004.

Zahedi, M., Shahin, M. and Ali, M. (2016) 'A systematic review of knowledge sharing challenges and practices in global software development', *International Journal of Information Management, Elsevier*. Elsevier Ltd, 36, pp. 995–1019.

Zanaty, F. El *et al.* (2018) 'An empirical study of design discussions in code review', in *Proc. 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement - ESEM '18*, pp. 1–10. doi: 10.1145/3239235.3239525.

Zhang, Z. X. (2001) 'The effects of frequency of social interaction and relationship closeness on reward allocation', *Journal of Psychology: Interdisciplinary and Applied*, 135(2), pp. 154–164. doi: 10.1080/00223980109603687.

APPENDICES

Appendix A Search Strings Executed to Database

An example of search strings executed to ACM database

Data base	Search String	Total Papers Found	1st level Extraction	2nd level Extraction	3rd level Extraction	Papers before Quality Assessment
ACM	(knowledge sharing OR knowledge transfer) AND (modern code review) AND (software engineering waiting waste)	32	6	14	00	12
	(knowledge sharing or knowledge transfer) AND (modern code review or contemporary code review) AND (lean software development OR lean software engineering)	26	9	13	00	04
	(knowledge sharing) AND (modern code review) AND (software engineering linger waste OR software engineering delay waste)	19	10	08	00	01
	(knowledge sharing) AND (modern code review) AND (software engineering blocking waste)	21	13	05	00	03
	(knowledge sharing) AND (modern code review) AND (software development delay waste)	157	153	00	01	03
	(knowledge sharing) AND (modern code review) AND (software development linger waste)	121	118	02	00	01
	(knowledge sharing) AND (modern code inspection) AND (lean software engineering OR lean software development)	264	257	07	07	00
	(knowledge sharing) AND (code review) AND (lean software engineering)	301	294	05	01	01
	(knowledge sharing) AND code inspection) AND (lean software engineering OR lean software development)	1017	996	10	09	02

Data base	Search String	Total Papers Found	1st level Extraction	2nd level Extraction	3rd level Extraction	Papers before Quality Assessment
	knowledge sharing) AND (lightweight code review) AND (lean software engineering OR lean software development)	261	253	07	00	01
	(knowledge sharing) AND (lightweight code inspection (lean software engineering OR lean software development)	121	115	05	00	01
	(knowledge sharing) AND (peer code review) AND (lean software engineering OR lean software development)	266	246	15	00	05
	(knowledge dissemination) AND (modern code review) AND (lean software engineering)	776	765	11	11	00
	(knowledge exchange) modern code review) AND (lean software engineering)	827	818	07	00	02
	(knowledge exchange OR knowledge transfer) AND (modern code review) AND (lean software engineering OR lean software engineering)	610	540	59	06	05
	(knowledge exchange) AND (contemporary code review) AND (lean software engineering)	715	703	12	02	10

Appendix B Distribution of Data Sources

Distribution of data sources for particular database

Database Repository	Papers Found	Inclusion/ Exclusion Criteria	Papers Selected after Inclusion and exclusion	Exclusion after QA	Papers Included for detail review after QA
ACM	2209	2151	58	1	57
IEEE	4052	3970	82	5	77
Springer Link	1420	1409	11	0	11
Wiley Online	516	514	2	0	2
Scopus	804	801	3	0	3
Web-of Science	288	282	6	0	6
Total Research Paper	**9289**	**9127**	**162**	**6**	**156**

Appendix C Quality Assessment Scores of Research Papers

Quality assessment scores of selected papers

Paper ID	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Cumulative Quality Assessment Score
KSFP1	Y	Y	Y	Y	Y	Y	Y	7
KSFP2	Y	Y	Y	Y	Y	Y	Y	7
KSFP3	Y	Y	Y	Y	Y	Y	Y	7
KSFP4	Y	Y	Y	Y	Y	Y	Y	7
KSFP5	Y	Y	Y	Y	Y	Y	P	6.5
KSFP6	Y	P	P	Y	Y	Y	P	5.5
KSFP7	Y	Y	Y	Y	Y	Y	Y	7
KSFP8	Y	Y	Y	Y	Y	Y	Y	7
KSFP9	Y	Y	Y	Y	Y	Y	Y	7
KSFP10	Y	Y	Y	Y	Y	Y	Y	7
KSFP11	Y	P	Y	Y	P	Y	P	5.5
KSFP12	Y	Y	Y	Y	Y	P	P	6
KSFP13	Y	Y	Y	P	Y	Y	P	6
KSFP14	Y	P	Y	Y	Y	Y	P	6
KSFP15	Y	Y	Y	P	Y	Y	Y	6.5
KSFP16	Y	Y	Y	Y	P	Y	P	6
KSFP17	Y	Y	Y	Y	P	Y	P	6
KSFP18	Y	P	Y	Y	Y	Y	Y	6.5
KSFP19	Y	Y	P	Y	N	Y	P	5
KSFP20	Y	Y	Y	P	Y	Y	Y	6.5
KSFP21	Y	Y	Y	Y	Y	Y	Y	7
KSFP22	Y	Y	Y	Y	Y	Y	Y	7
KSFP23	Y	Y	Y	P	Y	Y	P	6
KSFP24	Y	Y	P	P	Y	Y	P	5.5
KSFP25	Y	Y	Y	Y	Y	Y	P	6.5
KSFP26	Y	Y	Y	P	P	P	P	5
KSFP27	Y	Y	Y	Y	Y	Y	Y	7
KSFP28	Y	Y	Y	Y	Y	Y	Y	7
KSFP29	Y	Y	Y	0	Y	P	P	5
KSFP30	Y	P	Y	Y	Y	P	P	5.5
KSFP31	Y	Y	Y	P	P	P	P	5
KSFP32	Y	Y	P	P	P	Y	P	5
KSFP33	Y	P	Y	Y	Y	P	Y	6
KSFP34	Y	Y	Y	Y	Y	P	P	6
KSFP35	Y	Y	Y	P	Y	Y	P	6
KSFP36	Y	Y	Y	Y	P	Y	P	6
KSFP37	Y	Y	Y	Y	Y	Y	Y	7
KSFP38	Y	Y	Y	Y	Y	Y	Y	7
KSFP39	Y	Y	Y	P	Y	Y	Y	6.5
KSFP40	Y	Y	Y	Y	Y	Y	Y	7

Paper ID	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Cumulative Quality Assessment Score
KSFP41	Y	Y	Y	P	Y	P	Y	6
KSFP42	Y	Y	P	Y	P	P	P	5
KSFP43	Y	Y	Y	Y	P	Y	P	6
KSFP44	Y	Y	Y	Y	P	Y	P	6
KSFP45	Y	P	Y	P	Y	Y	P	5.5
KSFP46	Y	Y	Y	P	P	Y	P	5.5
KSFP47	Y	Y	Y	Y	Y	Y	P	6.5
KSFP48	Y	Y	Y	Y	Y	Y	Y	7
KSFP49	Y	Y	P	P	P	Y	P	5
KSFP50	Y	Y	Y	Y	P	Y	Y	6.5
KSFP51	Y	Y	Y	P	Y	Y	P	6
KSFP52	Y	Y	Y	Y	P	Y	P	6
KSFP53	Y	Y	Y	Y	P	Y	P	6
KSFP54	Y	Y	Y	Y	P	Y	Y	6.5
KSFP55	Y	Y	Y	Y	P	Y	Y	6.5
KSFP56	Y	Y	Y	P	Y	Y	Y	6.5
KSFP57	Y	P	Y	P	Y	Y	Y	6
KSFP58	Y	Y	Y	P	Y	Y	P	6
KSFP59	Y	Y	Y	P	Y	Y	P	6
KSFP60	Y	Y	Y	Y	N	Y	P	5.5
KSFP61	Y	Y	Y	P	P	Y	P	5.5
KSFP62	Y	Y	Y	Y	P	Y	P	6
KSFP63	Y	Y	Y	Y	P	Y	P	6
KSFP64	Y	Y	Y	Y	Y	Y	P	6.5
KSFP65	Y	P	P	P	Y	Y	Y	5.5
KSFP66	Y	Y	Y	P	P	Y	P	5.5
KSFP67	Y	Y	Y	P	P	P	Y	5.5
KSFP68	Y	Y	Y	Y	Y	Y	P	6.5
KSFP69	Y	Y	Y	Y	P	Y	Y	6.5
KSFP70	Y	Y	Y	Y	P	Y	P	6
KSFP71	Y	Y	Y	Y	P	Y	Y	6.5
KSFP72	Y	Y	Y	Y	Y	Y	Y	7
KSFP73	Y	Y	Y	Y	Y	Y	Y	7
KSFP74	Y	P	Y	Y	Y	Y	P	6
KSFP75	Y	Y	Y	Y	Y	P	Y	6.5
KSFP76	Y	Y	Y	Y	Y	Y	P	6.5
KSFP77	Y	Y	Y	Y	Y	Y	Y	7
KSFP78	Y	P	Y	Y	Y	Y	P	6
KSFP79	Y	Y	Y	P	Y	Y	P	6
KSFP80	Y	Y	Y	P	Y	P	Y	6
KSFP81	Y	Y	P	P	Y	P	P	5
KSFP82	Y	Y	Y	Y	P	Y	P	6
KSFP83	Y	Y	Y	P	Y	P	Y	6
KSFP84	Y	Y	P	Y	P	P	P	5

Paper ID	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Cumulative Quality Assessment Score
KSFP85	Y	P	Y	Y	Y	Y	P	6
KSFP86	Y	Y	Y	Y	Y	Y	Y	7
KSFP87	Y	Y	Y	P	Y	Y	P	6
KSFP88	Y	P	P	Y	Y	P	Y	5.5
KSFP89	Y	Y	Y	Y	Y	Y	P	6.5
KSFP90	Y	Y	P	Y	Y	Y	P	6
KSFP91	Y	Y	Y	P	Y	Y	Y	6.5
KSFP92	Y	Y	Y	P	Y	Y	P	6
KSFP93	Y	Y	P	Y	Y	P	P	5.5
KSFP94	Y	Y	P	Y	Y	Y	P	6
KSFP95	Y	Y	P	Y	Y	P	Y	6
KSFP96	Y	P	Y	P	Y	Y	Y	6
KSFP97	Y	Y	P	Y	Y	Y	Y	6.5
KSFP98	Y	Y	Y	Y	Y	Y	P	6.5
KSFP99	Y	Y	Y	Y	Y	Y	Y	7
KSFP10	Y	Y	Y	P	Y	Y	Y	6.5
KSFP10	Y	Y	P	Y	Y	Y	P	6
KSFP10	Y	Y	P	Y	Y	Y	P	6
KSFP10	Y	Y	Y	Y	Y	Y	Y	7
KSFP10	Y	Y	Y	Y	Y	Y	Y	7
KSFP10	Y	Y	P	P	Y	Y	Y	6
KSFP10	Y	Y	Y	Y	Y	Y	Y	7
KSFP10	Y	Y	P	P	Y	Y	Y	6
KSF108	Y	Y	P	P	Y	P	Y	5.5
KSFP10	Y	P	P	Y	Y	Y	Y	6
KSFP11	Y	Y	Y	P	Y	Y	Y	6.5
KSFP11	Y	P	Y	Y	P	Y	Y	6
KSFP11	Y	Y	P	P	Y	Y	Y	6
KSFP11	Y	Y	P	P	Y	Y	Y	6
KSFP11	Y	Y	Y	Y	Y	Y	Y	7
KSFP11	Y	Y	P	P	Y	Y	Y	6
KSFP11	Y	Y	Y	Y	Y	Y	Y	7
KSFP11	Y	Y	Y	Y	Y	Y	Y	7
KSFP11	Y	Y	Y	P	Y	Y	P	6
KSFP11	Y	Y	Y	P	Y	Y	Y	6.5
KSFP12	Y	Y	Y	Y	Y	Y	Y	7
KSFP12	Y	Y	Y	P	Y	Y	Y	6.5
KSFP12	Y	Y	Y	P	Y	Y	Y	6.5
KSFP12	Y	Y	Y	P	Y	Y	Y	6.5
KSFP12	Y	P	Y	Y	Y	Y	P	6
KSFP12	Y	Y	Y	Y	Y	P	Y	6.5
KSFP12	Y	Y	Y	Y	Y	Y	Y	7
KSFP12	Y	Y	Y	Y	Y	P	Y	6.5
KSFP12	Y	Y	Y	P	Y	Y	Y	6.5

Paper ID	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Cumulative Quality Assessment Score
KSFP12	Y	Y	Y	Y	Y	P	P	6
KSFP13	Y	Y	Y	P	Y	Y	Y	6.5
KSFP13	Y	P	Y	Y	Y	Y	P	6
KSFP13	Y	Y	Y	P	Y	Y	P	6
KSFP13	Y	Y	Y	Y	Y	Y	Y	7
KSFP13	Y	Y	Y	Y	Y	Y	Y	7
KSFP13	Y	Y	P	P	Y	P	Y	5.5
KSFP13	Y	Y	P	P	Y	Y	Y	6
KSFP13	Y	Y	Y	Y	Y	Y	Y	7
KSFP13	Y	Y	Y	Y	Y	Y	Y	7
KSFP13	Y	Y	Y	P	Y	Y	Y	6.5
KSFP14	Y	Y	Y	Y	Y	Y	P	6.5
KSFP14	Y	Y	Y	Y	Y	Y	Y	7
KSFP14	Y	Y	Y	Y	Y	Y	P	6.5
KSFP14	Y	Y	Y	Y	Y	Y	P	6.5
KSFP14	Y	Y	Y	Y	Y	Y	Y	7
KSFP14	Y	Y	P	P	Y	Y	Y	6
KSFP14	Y	Y	Y	Y	Y	Y	P	6.5
KSFP14	Y	Y	Y	Y	Y	Y	Y	7
KSFP14	Y	Y	Y	P	Y	Y	Y	6.5
KSFP14	Y	Y	Y	P	Y	Y	Y	6.5
KSFP15	Y	Y	Y	Y	Y	Y	Y	7
KSFP15	Y	Y	Y	Y	Y	Y	Y	7
KSFP15	Y	Y	Y	Y	Y	Y	Y	7
KSFP15	Y	Y	Y	Y	Y	Y	P	6.5
KSFP15	Y	Y	Y	Y	Y	P	Y	6.5
KSFP15	Y	Y	Y	P	Y	Y	Y	6.5
KSFP15	Y	Y	Y	Y	Y	P	P	6

Quality assessment scores of excluded papers

Paper Title	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Cumulative Quality Assessment Score
Collaborations and Code Reviews	Y	P	P	P	P	Y	P	4.5
How long does it take to fix the code: A case study of Open Stack	Y	Y	P	P	P	P	P	4.5

Paper Title	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Cumulative Quality Assessment Score
Gamifying software engineering tasks based on cognitive principles: The case of code review	Y	P	P	P	P	P	P	4
MCT: A Tool for Commenting Programs by Multimedia Comments	Y	Y	P	0	P	P	Y	4.5
Does Bug Prediction Support Human Developers? Findings from a Google Case Study	Y	Y	P	P	P	P	P	4.5
0-1 Programming Model-Based Method for Planning Code Review using Bug Fix History	Y	Y	Y	P	N	P	P	4.5

Appendix D Research Papers Selected for SLR after Quality Assessment

Research studies selected for SLR after quality assessment

Paper ID	Paper Title
KSFP1	Code reviewing in the trenches challenges and best practices
KSFP2	Code review quality: how developers see it?
KSFP3	Was my contribution fairly reviewed?" a framework to study the perception of fairness in modern code reviews
KSFP4	The effect of poor source code lexicon and readability on developers' cognitive load
KSFP5	Understanding review expertise of developers: a reviewer recommendation approach based on latent Dirichlet allocation
KSFP6	Poster: understanding and leveraging developer inexpertise
KSFP7	Studying pull request merges: a case study of shopify's active merchant
KSFP8	What makes a code change easier to review? an empirical investigation on code change reviewability
KSFP9	Modern code review: a case study at google
KSFP10	Comparing sequential and parallel code review techniques
KSFP11	An empirical study of design discussions in code review
KSFP12	BLIMP tracer: integrating build impact analysis with code review
KSFP13	The impact of human factors on the participation decision of reviewers in modern code review
KSFP14	Profile based recommendation of code reviewers
KSFP15	Communicative intention in code review questions
KSFP16	Context is king: the developer perspective on the usage of static analysis tools
KSFP17	Are fix-inducing changes a moving target? a longitudinal case study of just-in-time defect prediction
KSFP18	Information needs in contemporary code review
KSFP19	Code review tool for visual programming languages
KSFP20	Code review comments: language matters
KSFP21	Analysing the impact of feedback in GitHub on the software developer's mood
KSFP22	Review feedbacks influence to a contributor's time spent on OSS projects?
KSFP23	Feedback topics in modern code review: automatic identification and impact on changes
KSFP24	Codeflow: improving the code review process at Microsoft
KSFP25	CFAR: a tool to increase communication, productivity, and review quality in collaborative code review
KSFP26	Visualization of inter-module dataflow through global variables for source code review
KSFP27	Does reviewer recommendation help developers?

Paper ID	Paper Title
KSFP28	When testing meets code review: why and how developers review tests
KSFP29	CROP: linking code reviews to source code changes
KSFP30	Assisted discovery of software vulnerabilities
KSFP31	Salient-class location: help developers understand code change in code review
KSFP32	Poster: guiding developers to make informative commenting decisions in source code
KSFP33	State of mutation testing at google
KSFP34	Eye movements in code review
KSFP35	Finding impact factors for rejection of pull requests on GitHub
KSFP36	A large-scale study of test coverage evolution
KSFP37	Investigating the effectiveness of peer code review in distributed software development
KSFP38	Process aspects and social dynamics of contemporary code review: insights from open source development as well as industrial practice at Microsoft
KSFP39	Code review analysis of software system using machine learning techniques harsh
KSFP40	Continuous code reviews a social coding tool for code reviews inside the IDE
KSFP41	Broadcast vs. unicast review technology: does it matter?
KSFP42	Impact of continuous integration on code reviews
KSFP43	Comparing pre-commit reviews and post-commit reviews using process simulation
KSFP44	Confusion detection in code reviews
KSFP45	Evaluating how static analysis tools can reduce code review effort
KSFP46	A large-scale study of modern code review and security in open source projects
KSFP47	A hybrid approach to code reviewer recommendation with collaborative filtering
KSFP48	Understanding the impressions, motivations, and barriers of onetime code contributors to floss projects: a survey
KSFP49	The top 10 adages in continuous deployment
KSFP50	What are they talking about? analysing code reviews in pull-based development model article
KSFP51	Are fix-inducing changes a moving target? a longitudinal case study of just-in-time defect prediction
KSFP52	Decoding the representation of code in the brain: an FMRI study of code review and expertise
KSFP53	Who should comment on this pull request? analysing attributes for more accurate commenter recommendation in pull-based development
KSFP54	Search-based peer reviewers' recommendation in modern code review
KSFP55	Review participation in modern code review. An empirical study of the android, qt, and open stack projects
KSFP56	On the optimal order of reading source code changes for review
KSFP57	Experimental validation of source code reviews on mobile devices
KSFP58	Using metrics to track code review performance

Paper ID	Paper Title
KSFP59	WAP: does reviewer age affect code review performance?
KSFP60	How is if statement fixed through code review? a case study of QT project
KSFP61	An empirical study of reviewer recommendation in pull-based development model
KSFP62	The impact of continuous integration on other software development practices: a large-scale empirical study
KSFP63	Interactively decomposing composite changes to support code review and regression testing
KSFP64	Predicting usefulness of code review comments using textual features and developer experience
KSFP65	Which review feedback did long-term contributors get on OSS projects?
KSFP66	Semantics-assisted code review an efficient toolchain and a user study
KSFP67	Refactoring-aware code review: a systematic mapping study
KSFP68	SENTICR: a customized sentiment analysis tool for code review interactions
KSFP69	Characterizing software engineering work with personas based on knowledge worker actions
KSFP70	Are one-time contributors different? a comparison to core and periphery developers in floss repositories
KSFP71	Reviewer recommendation for pull-requests in GitHub: what can we learn from code review and bug assignment?
KSFP72	Work practices and challenges in pull-based development: the contributor's perspective
KSFP73	Factors influencing code review processes in industry
KSFP74	A collaborative code review platform for GitHub
KSFP75	A study of the quality-impacting practices of modern code review at Sony mobile
KSFP76	A security perspective on code review: the case of chromium
KSFP77	A faceted classification scheme for change-based industrial code review processes
KSFP78	Code review participation: game theoretical modelling of reviewers in Gerrit datasets
KSFP79	Revisiting code ownership and its relationship with software quality in the scope of modern code review
KSFP80	Quantifying and mitigating turnover-induced knowledge loss: case studies of chrome and a project at AVAYA
KSFP81	Peer review social network (PeRSoN) in open source projects
KSFP82	Mining the modern code review repositories: a dataset of people, process, and product
KSFP83	The emotional side of software developers in JIRA
KSFP84	Visualizing code and coverage changes for code review
KSFP85	Automatically recommending code reviewers based on their expertise: an empirical comparison
KSFP86	Correct: code reviewer recommendation at GitHub for vendasta technologies
KSFP87	Predicting defectiveness of software patches

Paper ID	Paper Title
KSFP88	Effective assignment and assistance to software developers and reviewers
KSFP89	Characterization of the xen project code review process: an experience report
KSFP90	Teaching code review management using branch based workflows
KSFP91	Automatically recommending peer reviewers in modern code review
KSFP92	Who should review this change? putting text and file location analyses together for more accurate recommendations
KSFP93	Lessons learned from building and deploying a code review analytics platform
KSFP94	Code reviews do not find bugs. how the current code review best practice slows us down
KSFP95	Code review: Veni, Vidi, Vici
KSFP96	Why did this reviewed code crash? an empirical study of Mozilla Firefox
KSFP97	Do code review practices impact design quality? a case study of the QT, VTK, and ITK projects
KSFP98	Interactive code review for systematic changes
KSFP99	Characteristics of useful code reviews: an empirical study at Microsoft
KSFP100	Investigating code review quality: do people and participation matter?
KSFP101	Four eyes are better than two: on the impact of code reviews on software quality
KSFP102	Partitioning composite code changes to facilitate code review
KSFP103	Investigating technical and non-technical factors influencing modern code review
KSFP104	Wait for it: determinants of pull request evaluation latency on GitHub
KSFP105	Helping developers help themselves: automatic decomposition of code review changesets
KSFP106	Investigating code review practices in defective files: an empirical study of the QT system,
KSFP107	Would static analysis tools help developers with code reviews?
KSF108	An exploratory study to identify similar patches: a case study in modern code review
KSFP109	Developers assignment for analysing pull requests
KSFP110	Network structure of social coding in GitHub
KSFP111	CoreDevRec: automatic core member recommendation for contribution evaluation
KSFP112	Treating software quality as a first-class entity
KSFP113	Vidi: the visual design inspector
KSFP114	Will they like this? evaluating code contributions with language models
KSFP115	Let's talk about it: evaluating contributions through discussion in GitHub
KSFP116	The impact of code review coverage and code review participation on software quality a case study of the QT, VTK, and ITK projects
KSFP117	Modern code reviews in open-source projects: which problems do they fix?
KSFP118	Peer review on open-source software projects: parameters, statistical models, and theory
KSFP119	Who does what during a code review? datasets of OSS peer review repositories

Paper ID	Paper Title
KSFP120	Peer impressions in open source organizations: a survey
KSFP121	Influence of social and technical factors for evaluating contribution in GitHub
KSFP122	Impact of developer reputation on code review outcomes in OSS projects. an empirical investigation
KSFP123	How do social interaction networks influence peer impressions formation? a case study
KSFP124	An empirical investigation of socio-technical code review metrics and security vulnerabilities
KSFP125	Understanding review helpfulness as a function of reviewer reputation, review rating, and review depth
KSFP126	Identifying the characteristics of vulnerable code changes: an empirical study
KSFP127	Security and emotion: sentiment analysis of security discussions on GitHub
KSFP128	Tracing back the history of commits in low-tech reviewing environments a case study of the Linux kernel
KSFP129	RefDistiller: a refactoring aware code review tool for inspecting manual refactoring edits
KSFP130	towards refactoring-aware code review
KSFP131	Code review analytics: Webkit as case study
KSFP132	Mining peer code review system for computing effort and contribution metrics for patch reviewers
KSFP133	Reviewer recommender of pull-requests in GitHub
KSFP134	Reviewer recommendation to expedite crowd collaboration
KSFP135	Critics: an interactive code review tool for searching and inspecting systematic changes
KSFP136	Writing acceptable patches: an empirical study of open source project patches
KSFP137	Convergent contemporary software peer review practices
KSFP138	Expectations, outcomes, and challenges of modern code review
KSFP139	Impact of peer code review on peer impression formation: a survey
KSFP140	Impression formation in online peer production: activity traces and personal profiles in GitHub
KSFP141	Will my patch make it? and how fast? case study on the Linux kernel
KSFP142	Reducing human effort and improving quality in peer code reviews using automatic static analysis and reviewer recommendation
KSFP143	The influence of non-technical factors on code review
KSFP144	Code review for newcomers: Is it different?
KSFP145	Gerrit software code review data from android
KSFP146	Assessing MCR discussion usefulness using semantic similarity
KSFP147	When a patch goes bad: exploring the properties of vulnerability-contributing commits
KSFP148	A study on the interplay between pull request review and continuous integration builds
KSFP149	An empirical study on the effectiveness of security code review

Paper ID	Paper Title
KSFP150	On the understanding of programs with continuous code reviews
KSFP151	Confusion in code reviews: reasons, impacts, and coping strategies
KSFP152	Social network site skills for communication professionals: conceptualization, operationalization, and an empirical investigation
KSFP153	Associating working memory capacity and code change ordering with code review performance
KSFP154	Expressions of sentiments during code reviews: male vs. female
KSFP155	Investigating the social representations of code smell identification: a preliminary study
KSFP156	Decomposing composite changes for code review and regression test selection in evolving software

Appendix E Implementation of Data Coding Techniques

Examples of implementation of data coding techniques within data source KSFP1

Paper Statement	Open Coding	Focused Coding	Axial Coding
[1] *"Interestingly, not all teams have* [1a]*explicit rules or* [1b]*policies around code review and* [1c]*code review policy vary".*	[1a] Team Rules [1b]Team Policies [1c]Variation in Code Review Policy	**Team Strategies** - Team Rules -Team Policies -Team Workflow -Variation in Code Review Policies	▲**Team** ➔ Team Strategies ➔ Team Culture ➔ Team Intentions ➔ Team Drives ➔ Team Organization
[2] "[2a]*Iteration involving* [2b]*communication between authors and reviewers".*	[2a]Iteration [2b]Communication		
[3] *"Notification of the selected reviewers as well as other stakeholders, with team policy dictating who should be informed and how".*	[3]Team Policy for Notification of Reviewer		
4 "the [4a]*order of review steps can vary slightly depending on a* [4b]*team's policies,* [4c]*culture, and* [4d]*tools".*	[4a]Order of Review Steps [4b]Team Policies [4c]Team Culture [4d]Review Tools	**Team Culture** **Team Intentions** -Improve Code -Finding Defects -Transfer Knowledge -Explore Alternative Solution -Improve Development Process -Avoid Build Breaking -Increase Team Awareness -Share Code Ownership -Assess Team	

Paper Statement	Open Coding	Focused Coding	Axial Coding
"Whether they are a code author or reviewer, the [5]process also helps them become more confident".	[5]Code Review Process	**Process** -Code Review Process **Tool** -Review Tool	▲**Facility Conditions** ➔ Process ➔ Tool ➔ Communication ➔ Organization Support
[6]"Most communication between author and reviewer occurs through the [6a]code review tool, but other [6b]communication channels, such as [6c]face-to-face discussions, [6d]whiteboard sessions, [6e]video and [6f] voice chats, are used for contentious issue".	[6a]Communication through Code Review Tool [6b]Face to Face Discussion [6c]White board Session [6d]Video Chats [6e]Voice Chats	**Communication** -Communication Channel	
[7]"Microsoft Engineers perform code reviews [7a]to improve code, [7b]find defects, [7c] transfer knowledge, [7d]explore alternative solutions [7e]improve the development process [7f]avoid build breaks, [7g]increase team awareness [7h]share code ownership, [7i] to assess the team".	[7a] Improve Code [7b]Finding Defects [7c]Transfer Knowledge [7d]Explore Alternative Solution [7e]Improve Development Process [7f]Avoid Build Break [7g]Increase Team Awareness [7h]Share Code Ownership [7i] Assess Team		
[8]"getting timely feedback as their top challenge".	[8]Feedback Timeliness	**Feedback** -Feedback Temporal Aspect -Feedback Usefulness	▲**Artefact** ➔ Feedback ➔ Source Code ➔ Testing

Paper Statement	Open Coding	Focused Coding	Axial Coding
[9] *"Usually you write up some code and then you send it out for review, and then about a 9a day later you ping them to remind them... and then about half a day later you go to their office and knock on their door".*	[9]Delay Feedback		
[10] *"reviewers sometimes focus on insignificant details rather than looking for larger issues"*	[10]Insignificant Details		
[11] *"There is a lot of style [comments] a lot of the time, which I find annoying. And people will be like, maybe you should use this name?"*	[11]Style Comments		
[12] *"When preparing for a review, interviewees said they are unsure how to document changes for review".*	[12]Change Documentation	**Source Code** -Change Documentation -Source Code Complexity -Source Code Structure	
[13] *"tooling slows down code velocity and [13a]tools should be modified to better suit the [13b]team's context, [13c]workflow, and [13d]policies".*	[13a]Review Tool [13b]Team Context [13c]Team Workflow [13d]Team Policies	**Team Organization** -Team Context -Team Size	
[14] *"receiving a [14a]rejection can be harsh and that they prefer being given a [14b]reason why a change is rejected".*	[14a]Rejection [14b]Convey Rejection Reason	**Individual Emotions** -Fear -Frustration	
[15] *"it can be tough managing multiple communication channels".*	[15]Communication Channel		
[16] *"Code reviewers said they struggle with large reviews".*	[16] Review Size		

Paper Statement	Open Coding	Focused Coding	Axial Coding
[17] *"understanding the* [17a] *code's purpose, the* [17b] *motivations for the change, and* [17c] *how the change was implemented".*	[17a] Code Purpose [17b] Change motivation [17c] Change Implementation Procedure		
[18] *"For code changes that are* [18a] *large and* [18b] *difficult to understand, one developer expressed* [18c] *frustration around the value of his review: "It's just this big incomprehensible mess... then you can't add any value because they are just going to explain it to you and you're going to parrot back what they say".*	[18a] Change Size [18b] Complex Change [18c] Frustration		
"*Regarding Comprehension, finding relevant* [19] *documentation about changes was another frequently reported challenge".*	[19] Change Documentation		
"A lack of [20] *training on the review process itself, and that their reviewing activities are perceived as not being valued enough".*	[20] Review Process Activities		
[21] "*lack insights into how their code review activities impact job evaluations".*	[21] Reviewer Awareness Impact of Code Review on Job	**<u>Individual Awareness</u>** -Awareness of Role-oriented Task	▲**<u>Individual</u>** ➔ **Individual Awareness** ➔ **Individual Historical Factors** ➔ **Individual Intention** ➔ **Individual Emotions**

Paper Statement	Open Coding	Focused Coding	Axial Coding
[22]*"when authors prepare a change for review, they should read through the change thoroughly".*	[22]Pre-review the Change before Sending for Review		
[23]*"Viewing changes in a code review tool can expose simple issues (such as code style) to the author."*	[23]Pre-review by Author using Tool for Code Style.		
[24]"*Small, incremental changes that are be easier to understand".*	[24]Change Size		
[25]"[25a]*clustering related changes,* [25b]*documenting the motivation for a change, and* [25c]*describing the change and how to approach the review will help reviewers."*	[25a]Clustering of Related Changes [25b]Change Motivation [25c]Change Description		
[26]*"Authors should* [26a]*test their changes, and* [26b]*if no test exists, they should create one".*	[26a]Prior Testing of Changes [26b]Test Case	**Testing** -Test Case -Automated Testing -Manual Testing	
[27]*"Running automated analysis tools can expose formatting and low-level issues that would otherwise waste reviewers' time".*	[27]Automated Testing		
[28]*"authors should carefully consider* [28a]*when to skip a review while referring to their* [28b]*organization's code review policy (if one exists)".*	[28a]Decision to skip review [28b]Organizations Code review policy	**Organization Support** -Organization Strategies and Policies -Organization Tasks	
[29]"*they must determine* [29a]*how many reviewers are needed, consulting their* [9b]*organization's policy if necessary".*	[29a]Team Size [29b]*Organization's Policy*	**Team Organization** -Team Size	

Paper Statement	Open Coding	Focused Coding	Axial Coding
[30]*"It is important to select appropriate reviewers, Authors might select reviewers who have* [30a]*code expertise, are responsible for the code, or need* [30b]*to build expertise. If not against a* [30c]*team policy, it may be advisable to allow reviewers to volunteer for motivational reasons".*	[30a]Reviewer Expertise [30b] Build Expertise [30c] Team Policy	**Individual Historical Factor** -Individual Expertise **Individual Intention** -Build Expertise	
[31]*"reducing the senior engineers' load was an important consideration".*	[31]Individual Workload	**Individual Pressure** -Individual Workload	
[32]*"Reviewers should choose* [32a]*communication channels carefully. Richer channels, such as* [32b]*face-to-face or* [32c]*voice, are preferred for contentious issues or for discussing complex code changes. While for non-contentious or sensitive issues,* [3d]*tools that provide* [3e]*traceability are preferred".*	[32a]Communication Channels [32b]Face to Face [32c]Voice [32d]Review Tool [3e]Traceability Facility of tool	**Tool Support** -Automated Feature Assistance -Integration with Development Tool	
[33]*"skill to give* [33a]*constructive and* [33b]*respectful feedback while also clearly explaining the* [3c]*reasons for rejecting a change".*	[33a]Constructive Feedback [33b]Respectful Feedback [33c]Convey Reason for Rejection	**Feedback** -Feedback Structure	
[34]*"an organization should consider* [34a]*establishing a code review policy. Such a policy should help in building a* [34b]*positive review culture that sets the tone for* [34c]*constructive feedback"*	[34a]Establishment of Code Review Policy [34b]Positive Review Culture Policy [34c]Constructive Review Feedback	**Organization Support** -Organization Strategies and Policies	

Paper Statement	Open Coding	Focused Coding	Axial Coding
[35] *"organization or team should watch for negative impacts of* [35a]*employee assessment or* [35b]*incentives that may be linked to* [35c]*code reviewing activities".*	[35a]Employee Assessment [35b]Incentives [35c]Code Reviewing Activities		
[36] *"Encourage* [6a]*rewarding engineers who spend considerable effort reviewing others' code is encouraged,* [36b]*penalizing engineers who do not (often with a good reason) may lead to gaming of the system".*	[36a]Rewards [36b]Penalties		
[37] *"It is also important to ensure that author and reviewer use* [37a] *appropriate tools that match the desired* [37b]*reviewing culture and* [37c]*defined process (if there is one)".*	[37a]Review Tool [37b]Review Culture [37c]Review Process		
[38] *"Tools might support certain steps in the process, such as* [38a]*finding and* [38b]*notifying reviewers,* [38c]*automating feedback,* [38d]*running style checkers, and* [38e]*testing."*	[38a]Finding Reviewer Feature [38b]Notifying Reviewer Feature [38c]Automated Feedback [38d]Style Checker [38e]Automated Testing		
[39] *"Tools should be lightweight and* [39a]*integrate well with other developer tools, especially with* [39b]*informal* [39c]*communication channels."*	[39a]Integration of Review Tool with Development Tool [39b]Integration of Review Tool with Communication Channel [39c]Communication Channel		

Paper Statement	Open Coding	Focused Coding	Axial Coding
[40]*"Distributed teams might have additional tool needs".*	[40]Distributed Teams		
[41]*"knowing the expected* [41a]*process or how to use desired* [41b] *tools, an organization can ensure there is sufficient training in place* [41c]*Informal training through mentorship might be all that is required".*	[41a]Training of Process [41b]Training of Tool [41c]Informal Training		
[42]*"Finally, an organization should to* [41a]*develop,* [42b]*reflect on,* [42c]*revise code reviewing policies and checklists".*	[42a]Development of Code Review Policies and Checklist [42b]Reflect on Code Review Policies and Checklist [42c]Revision of Code Review Policies and Checklist	**Organization Support** -Organization Practices	

Examples of implementation of data coding techniques within data source KSFP2

Paper Statement	Open Coding	Focused Coding	Axial Coding
[1"]*developers with* [1a]*high workloads (i.e., over 10 patches/reviews per week) tend to concentrate their efforts on a single task type, i.e., either writing patches or reviewing them".*	[1]Developers' Workload	**Individual Pressure** -Individual Workload	
[2]*"The need for* [2a]*"dedicated" reviewers is pursued to bring their unique* [2b]*knowledge and* [2c]*expertise, e.g., overall* [2d]*architecture or* [2e]*domain knowledge, to the project to ensure the correctness and fit of code contributions".*	[2a]Dedicated Reviewers [2b]Reviewer Knowledge [2c]Reviewer Expertise [2d]Reviewer Architectural Knowledge [2e]Reviewer Domain Knowledge	**Individual Historical Factors** -Individual Characteristic -Individual Knowledge -Reviewer Expertise -Personality of the Reviewer	
[3"]*The majority of reviewers conduct code review in Bugzilla despite having access to a custom-built code review tool, and use various* [3a]*communication channels for discussing code modifications".*	[3a]Comunication Channel	**Communication Support** -Communication Channel	
[4"]*smaller* [4a]*patches are more likely to receive* [4b]*faster responses".*	[4a] Patch Size [4b]Response Time	**Source Code** -Complexity -Patch Size -Readability **Feedback** -Feedback Timeliness	

Paper Statement	Open Coding	Focused Coding	Axial Coding
[5]*"Readability/variable naming affecting how hard it is to understand any particular hunk of the patch on its own".*	[5]Readability		
[6]*"There are different characteristics that identify the suitability of a reviewer. For R87 it is "the [6a]personality of a reviewer", while for R52 it is presence of"[6b]personal backlog of work, and [6c]personal priorities".*	[6a]Personality of the Reviewer [6b]Personal Backlog of Work, [6c]Personal Priorities	**Individual Pressure** -Personal Backlog	
[7]*"When developers submit a patch they can include the [7a]results of running existing tests, as well as include the [7b]tests they wrote specifically for that patch".*	[7a]Test Results [7b]Test Case		
"The two sub- categories that we identified reflect the option patch writers have. The first sub-category is focused on the [8]presence of automated tests in a patch: "... changes that are accompanied by tests are much more likely to be accepted".	[8]Presence of Automated Test	**Testing** -Test Result -Test Case -Presence of Automated Test -Presence of Test	
[9]*"completeness of tests is also important: "thoroughness of tests included in patch".*	[9]Completeness of Test		
[10]*"including [10a]test results as a message on the bug tracker can either give the reviewer more confidence to accept the patch (if the tests pass) or likewise lead them to reject the patch (if the tests fail)".*	[10a] Test Results		

Paper Statement	Open Coding	Focused Coding	Axial Coding
[11] "[11a]*Change scope and* [11b]*rationale is believed to be an of influential factor for reviewers making their decisions".*	[11a]Change Scope [11b]Change Rationale		
[12] "*Developers believe that factors such as the* [12a]*experience of developers, the* [12b]*choice of a reviewer,* [12c]*size of a patch, its* [12d]*quality and* [12e]*rationale affect the time needed for review".*	[12a]Experience of Developers [12b]Choice of a Reviewer [12c]Size of a Patch [12d]Quality of Patch [12e]Patch rationale		
[13] "[13a]*bug severity,* [13b]*code quality and its* [13c]*rationale,* [3d]*presence and* [13e]*quality of tests, and* [13f]*developer personality impact review decisions".*	[13a]Bug Severity [13b]Code Quality [13c]Code Rationale [13d]Presence of Test [13e]Quality of Tests, [13f]Developers' Personality		
[14]"*Change rationale is the second top property that reviewers look for".*	[14]Change Rationale		
[15]"*Reviewers expect code changes to come with a* [15a]*corresponding test change. The lack of such tests is a good sign that "*[15b]*test coverage is lacking and we're taking a risk".*	[15a]Corresponding Test Change [15b]Test Coverage		
[16] "*The presence of tests in the patch also boosts developer's confidence".*	[16]Presence of Tests		

Paper Statement	Open Coding	Focused Coding	Axial Coding
[17] *"when testing is not practical, they perform* [17a]*manual testing as well. As a part of manual testing, developers often perform an* [17b]*operational proof such as code walks through".*	[17a]Manual Testing [17b]Operational Proof		
[18]*"*[18a]*clear and* [18b]*thorough feedback is the key attribute of a well-done re- view".*	[18a]Clear Feedback [18b]Thorough feedback		
[19]*"Reviewers are expected to provide feedback that is*[19a]*clear to understand; is not only "about* [19b]*code formatting and* [19c]*Style" (R6); 3) provides* [19d]*constructive advice".*	[19a]Clear Feedback [19b]Feedback Focusing Code Formatting [19c]Feedback Focusing Code Style [19d]Constructive Feedback		
[20] *"enough* [20a]*domain knowledge is always the first criteria for a well-done code".*	[20a]Domain Knowledge		
[21]*"personal factors such as* [21a]*patch writer experience,* [21b]*reviewer workloads,* [21c]*developer participation in the discussion of code changes, module and* [21d]*number of resubmitted patches are more likely to affect the quality of reviews".*	[21a]Patch Writer Experience [21b]Reviewer Workloads [21c]Developer Participation [21d]Number of Resubmitted Patches		
[22]*"Reviewers are often required to evaluate* [22]*large patches".*	[22] Patch Size		

Appendix F List of Knowledge Sharing Factors Attained after SLR

List of knowledge sharing factors, sub-factors, and categories attained after SLR with references

Category	KSF	KSF Paper ID	Sub-Factors	Sub-Factor Paper ID
Individual	**Individual Impartiality**	[KSFP1]	Biasness	[KSFP3, KSFP9, KSFP38, KSFP70, KSFP140]
			Balance Between Equity and Equality	[KSFP3, KSFP139]
	Individual Historical Factors	[KSFP120, KSFP140]	Individual Characteristics	[KSFP2, KSFP3, KSFP7, KSFP38, KSFP51, KSFP60, KSFP73, KSFP99, KSFP109]
			Individual Knowledge	[KSFP2, KSFP6, KSFP7, KSFP14, KSFP39, KSFP44, KSFP48, KSFP54, KSFP137, KSFP138, KSFP140, KSFP150]
			Individual Expertise	[KSFP1, KSFP2, KSFP3, KSFP5, KSFP6, KSFP17, KSFP18, KSFP19, KSFP38, KSFP47, KSFP51, KSFP52, KSFP54, KSFP55, KSFP69, KSFP71, KSFP73, KSFP77, KSFP78, KSFP92, KSFP145]
			Individual Experience	[KSFP1, KSFP2, KSFP3, KSFP6, KSFP7, KSFP8, KSFP9, KSFP13, KSFP17, KSFP27, KSFP32, KSFP37, KSFP48, KSFP54, KSFP55, KSFP59, KSFP68, KSFP71, KSFP73, KSFP77, KSFP78, KSFP124, KSFP141, KSFP149]
			Individual Technical Skills	[KSFP1, KSFP2, KSFP3, KSFP5, KSFP6, KSFP17, KSFP38, KSFP47, KSFP51, KSFP52, KSFP54, KSFP55, KSFP69, KSFP71, KSFP73, KSFP77, KSFP78, KSFP, KSFP85, KSFP86, KSFP92, KSFP140, KSFP145]
			Individual Non-Technical Skills	[KSFP2, KSFP9, KSFP38, KSFP43, KSFP48, KSFP73, KSFP146, KSFP152]
			Work Style	[KSFP1, KSFP7, KSFP8, KSFP46, KSFP69, KSFP77, KSFP116, KSFP120, KSFP140]

Category	KSF	KSF Paper ID	Sub-Factors	Sub-Factor Paper ID
			Work Track Record	[KSFP2, KSFP7, KSFP71]
			Affiliation	[KSFP7, KSFP8, KSFP38, KSFP140]
	Individual Emotions	[KSFP21, KSFP83, KSFP127, KSFP140, KSFP154, KSFP155]	Anger	[KSFP2, KSFP83, KSFP 127, KSFP151]
			Frustration	[KSFP2, KSFP83, KSFP124, KSFP127, KSFP140, KSFP151]
			Empathy	[KSFP2, KSFP72, KSFP83]
			Mood	[KSFP2, KSFP21, KSFP83, KSFP154]
			Fear	[KSFP1, KSFP3, KSFP38, KSFP72, KSFP73, KSF80, KSFP83, KSFP110, KSFP124, KSFP153]
	Individual Pressure	[KSFP2, KSFP3]	Cognitive Load	[KSFP4, KSFP9, KSFP66, KSFP153]
			Individual Workload	[KSFP1, KSFP2, KSFP3, KSFP7, KSFP8, KSFP13, KSFP39, KSFP47, KSFP54, KSFP136, KSFP140]
			Time Pressure	[KSFP1, KSFP2, KSFP7, KSFP8, KSFP12, KSFP38, KSFP39, KSFP47 , KSFP72, KSFP73, KSFP138, KSFP140, KSFP146]
			Context Switching	[KSFP2]
	Individual Awareness	[KSFP2, KSFP3]	Awareness of Code Quality	[KSFP1, KSFP2, KSFP4, KSFP7, KSFP8, KSFP38, KSFP39, KSFP41, KSFP47, KSFP66, KSFP79, KSFP94, KSFP120, KSFP137]
			Awareness of Process Improvement	[KSFP1, KSFP72, KSFP77, KSFP153]
			Awareness of Knowledge Sharing	[KSFP1, KSFP7, KSFP49, KSFP82, KSFP150]
			Awareness of Effective Communication	[KSFP2, KSFP9, KSFP38, KSFP71, KSFP72]

Category	KSF	KSF Paper ID	Sub-Factors	Sub-Factor Paper ID
			Awareness of Role-Oriented Tasks	[KSFP1, KSFP7, KSFP8, KSFP17, KSFP38, KSFP45, KSFP56, KSFP72, KSFP75, KSFP78, KSFP91, KSFP106, KSFP118, KSFP138]
	Individual Turnover	[KSFP80, KSFP127, KSFP150]	Job-Dissatisfaction	[KSFP80]
			Personal Conflicts	[KSFP39, KSFP80, KSFP127, KSFP140]
			Personal Issues	[KSFP80]
			Alternative Job Opportunities	[KSFP80]
			Impact of Turnover	[KSFP80]
	Individual Intentions	[KSFP1]	Self-Learning	[KSFP3, KSFP8, KSFP47, KSFP51, KSFP73, KSFP80, KSFP83, KSFP90, KSFP118]
			Collaboration	[KSFP38, KSFP75, KSFP82, KSFP137]
			Problem Solving	[KSFP9, KSFP38, KSFP75]
			Impression Formation	[KSFP2, KSFP38 , KSFP39 KSFP118, KSFP120, KSFP123 KSFP139, KSFP140 KSFP146]
			Build Relationships	[KSFP38, KSFP118]
Social	**Relational**	[KSFP10, KSFP110, KSFP121]	Trust	[KSFP2, KSFP7, KSFP8, KSFP9, KSFP38, KSFP47, KSFP53, KSFP54, KSFP71, KSFP75, KSFP106, KSFP111, KSFP120, KSFP121, KSFP123, KSFP137, KSFP139]
			Reputation	[KSFP2, KSFP7, KSFP38, KSFP48, KSFP104, KSFP122, KSFP125, KSFP136, KSFP140, KSFP141, KSFP178]
			Familiarity	[KSFP13, KSFP46, KSFP54, KSFP20, KSFP124]
			Frequency of Interaction	[KSFP39, KSFP50, KSFP77, KSFP121, KSFP140, KSFP144]

Category	KSF	KSF Paper ID	Sub-Factors	Sub-Factor Paper ID
	Structural	[KSFP81, KSFP144]	Social Network	[KSFP38, KSFP71, KSFP81, KSFP82, KSFP123, KSFP133, KSFP134, KSFP152]
			Social Network Ties	[KSFP104, KSFP121, KSFP134]
			Network Channel	[KSFP144, KSFP81]
			Network Stability	[KSFP123]
			Social Network Structure	[KSFP71, KSFP8, KSFP133]
			Socio-Political Structure	[KSFP72, KSFP124, KSFP140]
Artefact	**Source Code**	[KSFP1, KSFP2, KSFP38]	Source Code Structure	[KSFP1, KSFP2, KSFP3, KSFP4, KSFP7, KSFP8, KSFP9, KSFP17, KSFP13, KSFP14, KSFP16, KSFP23, KSFP26, KSFP37, KSFP38, KSFP40, KSFP41, KSFP42, KSFP44, KSFP54, KSFP55, KSFP57, KSFP76, KSFP82, KSFP87, KSFP116, KSFP132, KSFP137, KSFP143, KSFP153, KSFP155]
			Source Code Complexity	[KSFP2, KSFP3, KSFP7, KSFP8, KSFP13, KSFP31, KSFP34, KSFP37, KSFP38, KSFP54, KSFP58, KSFP63, KSFP70, KSFP 73, KSFP87, KSFP116, KSFP118, KSFP124, KSFP126, KSFP132, KSFP148, KSFP150, KSFP156]
			Source Code Readability	[KSFP2, KSFP4, KSFP8, KSFP9, KSFP10, KSFP32, KSFP33, KSFP34, KSFP38, KSFP41, KSFP137 , KSFP140 KSFP151]
			Source Code Efficiency	[KSFP2, KSFP5, KSFP10, KSFP43, KSFP47, KSFP151]
			Source Code Associated Risks	[KSFP2, KSFP3, KSFP18, KSFP38, KSFP126, KSFP128, KSFP140]
			Handling of Error Situations	[KSFP2, KSFP38]
			Adherence to Coding Standards	[KSFP8, KSFP10, KSFP14, KSFP16, KSFP45, KSFP59, KSFP60, KSFP72, KSFP137, KSFP140]
			Source Code Change Motivation	[KSFP1, KSFP2, KSFP44, KSFP72]

Category	KSF	KSF Paper ID	Sub-Factors	Sub-Factor Paper ID
			Source Code Change Documentation	[KSFP1, KSFP2, KSFP4, KSFP7, KSFP8, KSFP37, KSFP38, KSFP55, KSFP71, KSFP72 KSFP76, KSFP99, KSFP100, KSFP137, KSFP138]
			Source Code Change Scope	[KSFP2, KSFP7, KSFP8, KSFP117]
			Nature of Change	[KSFP8, KSFP37, KSFP128, KSFP137]
			Change Impact	[KSFP8, KSFP12, KSFP67, KSFP140, KSFP141, KSFP142]
			Source Code Change Revertability	[KSFP7]
	Feedback	[KSFP2, KSFP3, KSFP7]	Feedback Language	[KSFP3, KSFP7, KSFP10, KSFP19, KSFP20, KSFP38, KSFP39, KSFP110, KSFP154]
			Feedback Temporal Aspects	[KSFP1, KSFP3, KSFP7, KSFP8, KSFP9, KSFP13 KSFP38, KSFP39, KSFP 41, KSFP45, KSFP 47, KSFP53, KSFP54, KSFP51, KSFP70, KSFP72, KSFP78, KSFP91, KSFP94, KSFP103, KSFP118, KSFP122, KSFP131, KSFP140, KSFP141]
			Feedback Targeted Object	[KSFP3, KSFP10]
			Feedback Usefulness	[KSFP1, KSFP9, KSFP12, KSFP15, KSFP16, KSFP22, KSFP23, KSFP24, KSFP30, KSFP37, KSFP38, KSFP53, KSFP64, KSFP65, KSFP68, KSFP 90, KSFP94, KSFP100]
			Feedback Source	[KSFP1, KSFP37]
			Feedback Structure	[KSFP1, KSFP2, KSFP7, KSFP8, KSFP10, KSFP11, KSFP18, KSFP21, KSFP103, KSFP126]
			Feedback Training	[KSFP1, KSFP9, KSFP13, KSFP21, KSFP43, KSFP37, KSFP38, KSFP39, KSFP45, KSFP47, KSFP55, KSFP65, KSFP81, KSFP91, KSFP93, KSFP100, KSFP119, KSFP120]

Category	KSF	KSF Paper ID	Sub-Factors	Sub-Factor Paper ID
			Feedback Size	[KSFP9, KSFP13, KSFP37, KSFP38, KSFP39, KSFP43 KSFP45, KSFP47, KSFP55, KSFP65, KSFP81, KSFP91 , KSFP93, KSFP100, KSFP119, KSFP120, KSFP137]
			Feedback Cycle	[KSFP1, KSFP9, KSFP13, KSFP25, KSFP38, KSFP72, KSFP76, KSFP96, KSFP117]
			Feedback Content	[KSFP10, KSFP30 KSFP35, KSFP38, KSFP94, KSFP95, KSFP96]
			Feedback Perception	[KSFP10]
			Feedback Communication	[KSFP38, KSFP91, KSFP112, KSFP118, KSFP137]
			Feedback Frequency	[KSFP40, KSFP65, KSFP115, KSFP116, KSFP118, KSFP140]
			Defect Details Conveyed in Feedback	[KSFP10, KSFP38, KSFP95]
	Testing	[KSFP2, KSFP3, KSFP7]	Test Results	[KSFP2, KSFP136]
			Manual Tests	[KSFP2, KSFP3, KSFP72, KSFP149]
			Test Suits	[KSFP2, KSFP7, KSFP33, KSFP36, KSFP63, KSFP69, KSFP104]
			Test Quality	[KSFP7]
			Test Case	[KSFP1, KSFP2, KSFP8, KSFP63, KSFP115, KSFP121]
			Automated Tests	[KSFP1, KSFP2, KSFP7, KSFP9, KSFP28, KSFP36 , KSFP66, KSFP69, KSFP104]
			Test Documentation	[KSFP2, KSFP9, KSFP12, KSFP18, KSFP72, KSFP121]
			Test Coverage	[KSFP2, KSFP7, KSFP14, KSFP36, KSFP66, KSFP104]
			Test Type	[KSFP7]
			Proof of Testing	[KSFP2]

Category	KSF	KSF Paper ID	Sub-Factors	Sub-Factor Paper ID
Facility Conditions	**Process Support**	[KSFP2, KSFP3, KSFP7]	Development Process	[KSFP1, KSFP7, KSFP9, KSFP38, KSFP43, KSFP46, KSFP73]
			Review Process	[KSFP1, KSFP3, KSFP19, KSFP39, KSFP41, KSFP42, KSFP46, KSFP73, KSFP150]
			Process Complexity	[KSFP7, KSFP43, KSFP48, KSFP74, KSFP105, KSFP109, KSFP133]
			Process Selection	[KSFP1, KSFP3, KSFP73, KSFP84, KSFP101, KSFP108]
			Process Quality	[KSFP41, KSFP42]
			Process Availability	[KSFP74]
	Tool Support	[KSFP2, KSFP3, KSFP7, KSFP39, KSFP73]	Development Tool	[KSFP1, KSFP63]
			Review Tool	[KSFP3, KSFP17, KSFP19, KSFP25, KSFP26, KSFP28, KSFP29, KSFP67, KSFP88, KSFP111, KSFP112, KSFP 113, KSFP129, KSFP130, KSFP135]
			Technical Maturity	[KSFP1, KSFP2, KSFP9, KSFP74, KSFP84, KSFP138]
			Integration of Review Tool with Development Tool	[KSFP1, KSFP27, KSFP142]
			Automated Feature Assistance	[KSFP1, KSFP2, KSFP5, KSFP9, KSFP16, KSFP39, KSFP41, KSFP44, KSFP45, KSFP50, KSFP66, KSFP72, KSFP74, KSFP77, KSFP107, KSFP111, KSFP112, KSFP113, KSFP133, KSFP134, KSFP135]
			Selection of Tool	[KSFP2]
			Tool Flexibility	[KSFP1]
			Tool Complexity	[KSFP1, KSFP74]
			Tool Portability	[KSFP1, KSFP57]
			Tool Availability	[KSFP1, KSFP74]

Category	KSF	KSF Paper ID	Sub-Factors	Sub-Factor Paper ID
	Organization Support	[KSFP1, KSFP3, KSFP93]	Availability of Resources	[KSFP3, KSFP21, KSFP38, KSFP62, KSFP73]
			Organization Policies	[KSFP1, KSFP3, KSFP16, KSFP38, KSFP94, KSFP105, KSFP126]
			Organization Characteristics	[KSFP1, KSFP2, KSFP38, KSFP74, KSFP79, KSFP103]
			Organization Practices	[KSFP1, KSFP38, KSFP44, KSFP128, KSFP150]
	Communication Support	[KSFP2, KSFP25]	Communication Type	[KSFP1, KSFP2, KSFP140, KSFP141]
			Communication Channel	[KSFP1, KSFP2, KSFP7, KSFP112, KSFP24, KSFP25, KSFP44, KSFP72, KSFP75, KSFP77, KSFP120, KSFP121, KSFP137, KSFP139]
			Communication Purpose	[KSFP15, KSFP39, KSFP72]
			Communication Pattern	[KSFP2, KSFP38, KSFP39, KSFP73]
			Communication Procedure	[KSFP1]
	Project Support	[KSFP93]	Problem Domain	[KSFP2, KSFP4, KSFP7, KSFP16, KSFP17]
			Project Quality Assessment	[KSFP2]
			Project Attributes	[KSFP1, KSFP2, KSFP7, KSFP38, KSFP41, KSFP72, KSFP118, KSFP138, KSFP139]
			Release Management	[KSFP2, KSFP12, KSFP38, KSFP73]
			Adherence to Standards	[KSFP2, KSFP14, KSFP114, KSFP121, KSFP126]
			Risk Management	[KSFP3]
Team	**Team Organization**	[KSFP1]	Team Size	[KSFP3, KSFP116, KSFP65, KSFP121, KSFP124, KSFP131, KSFP140, KSFP149, KSFP150]
			Team Roles	[KSFP9, KSFP77, KSFP88, KSFP101, KSFP107, KSFP148]
			Team Responsibilities	[KSFP107, KSFP114, KSFP131]

Category	KSF	KSF Paper ID	Sub-Factors	Sub-Factor Paper ID
			Team Distance	[KSFP1, KSFP9, KSFP37, KSFP38, KSFP73, KSFP75, KSFP92, KSFP139]
			Role Multiplicity	[KSFP2]
	Team Strategies	[KSFP116]	Team Policies	[KSFP1, KSFP16, KSFP24, KSFP73, KSFP137]
			Team Work Practices	[KSFP1, KSFP24, KSFP38, KSFP72, KSFP90]
			Team Rules	[KSFP1, KSFP73, KSFP76]
			Team Work Processes	[KSFP1, KSFP90, KSFP136]
	Team Culture	[KSFP1, KSFP7, KSFP73]	Familiarity among Team Members	[KSFP21, KSFP124]
			Friction among Team Members	[KSFP9, KSFP72]
			Team Accountability	[KSFP100]
			Team Values	[KSFP9, KSFP73, KSFP155]
	Team Intentions	[KSFP3]	Identify Better Solutions	[KSFP38, KSFP59, KSFP137, KSFP138]
			Improve Code Quality	[KSFP2, KSFP79, KSFP138]
			Knowledge Distribution	[KSFP1, KSFP2, KSFP7, KSFP11 KSFP38, KSFP49, KSFP73, KSFP74, KSFP78, KSFP82, KSFP89, KSFP121]
			Improve Development Process	[KSFP1, KSFP7, KSFP153]
			Avoid Breaking Builds	[KSFP1]
			Share Code Ownership	[KSFP1, KSFP137]
			Increase Team Awareness	[KSFP1, KSFP59, KSFP74, KSFP78, KSFP150]
			Improve Software Quality	[KSFP54, KSFP76, KSFP79, KSFP80]
			Identify Defects	[KSFP1, KSFP38, KSFP54, KSFP73, KSFP137, KSFP138]

Category	KSF	KSF Paper ID	Sub-Factors	Sub-Factor Paper ID
	Team Drives	[KSFP1, KSFP72]	Team Productivity	[KSFP23, KSFP25, KSFP51, KSFP75, KSFP93, KSFP98, KSFP102, KSFP120, KSFP139]
			Team Motivations	[KSFP1, KSFP2, KSFP38]
			Team Priorities	[KSFP2, KSFP38, KSFP48, KSFP73, KSFP116, KSFP131]
			Team Workload	[KSFP1, KSFP2, KSFP3, KSFP7, KSFP13, KSFP39, KSFP54, KSFP136]
			Team Cohesion	[KSFP139]
			Team Participation	[KSFP1, KSFP7, KSFP12, KSFP13, KSFP17, KSFP38, KSFP61, KSFP89, KSFP97, KSFP101, KSFP108, KSFP109]

Appendix G Demographic Information

Experts of expert review

Expert Reviewer	Designation	Association	Experiance	Domain of Experience
Reviewer 1	Dean	-Superior Univeristy Lahore -Air University, Islamabad, Pakistan	15 years	-Software Engineering Teachning 13 years -Software Development & MCR 9 years
Reviewer 2	Dean	-My University, Islamabad, Pakistan -International Islamic University, Islamabad, Pakistan	15 years	-Software Engineering Professional teaching 12 years -Software Development 10 years
Reviewer 3	Assisatnt Professor	-International Islamic University, Islamabad Pakistan	12 years	-Software Engineering teachning 12 years -Software development 9 years
Reviewer 5	Product Manager	-Computer Share -LMKR, Islamabad, Pakistan	14 years	-Software Development 14 years -MCR 10 years

Delphi panel members

Delphi Expert ID	Designation	Association	Experience in Software Development	Domain of Experience
DP-01	Manager	Showroom, interwood mobel pvt ltd.	8 year	-Software development and MCR
DP-02	Software Developer	Broadlytech	8 years	-Software development and MCR
DP-03	Software Developer	Broadlytech	8 years	-Software development and MCR
DP-04	CEO	Broadlytech	08 years	-Software development and MCR
DP-05	Senior Web Developer	Quaid Tech	9 years	-Software development and MCR
DP-06	Pricipal Software Engineer	Datum Square Islamabad	9 Years	-Software development and MCR
DP-07	Software Developer	Broadlytech	9 years	-Software development and MCR
DP-08	Software Developer	Software Engineer Seven Technology Islsmsbad	09 years	-Software development and MCR
DP-09	Software Developer	Broadlytech	10 Years	-Software development and MCR
DP-10	Senior Software Engineering	Synergy IT	10 years	-Software development and MCR

Profile of Subjects Participated in Experiment

Subject ID	Program of Study	Programming Skills	Development Experience
1	BSCS	C++, Java	2 year 5 months
2	BSCS	C++, C, Java	2 year
3	BSCS	C++, C, Java , JavaScript	3 year
5	BSCS	C++, C, Java, PhP,	2 year
6	BSCS	C#, C++,	2 year 2 months
7	BSCS	C++, C#,	3 year
8	BSCS	C++, VB, C#	2 year
9	BSCS	C++, Java	3 year
10	BSCS	C++,	2.5 year
11	BSCS	C, Java, Visual C#	2 year 2 months
12	BSCS	C#, C++,	2 year
13	BSCS	C++, C#,	2 year
14	BSCS	C#, C++,	2 year
15	BSCS	C++, C#, VB	2 year
16	BSCS	Java, C++,	3 year 2 months
17	BSCS	C++,	2 year
18	BSCS	Visual Programming, C++, Java	2 year
19	BSCS	C++, C#, VB	2 year
20	BSCS	Java, C++,	3 year
21	BSCS	C, C++	2 year
22	BSCS	C++, C#, VB	2 year
23	BSCS	Java, C++, C	3 year
24	BSCS	C++,	2 year
25	BSCS	C#, C++,	2 year
26	BSCS	C++, C#, VB	2 year
27	BSCS	Java, C++,	3 year
28	BSCS	C++,	2 year

Appendix H Instructions and Feedback Form Template for Expert Review

Instructions

Review the naming conventions of factors, sub-factors and categories and please mention if there are any suggestions regarding naming convention in Column *"Suggestions on Naming Convention"* of the form.

Review the grouping and sub-grouping of knowledge sharing factors. Please mention the suggested modifications and names for the new and existing category/s in the column *"Suggestions on Grouping/Sub-grouping"* of Table, if required.

Suggesting new knowledge sharing factors that should be included in the list. Please mention the suggested knowledge sharing factors, sub-factors, or categories. in column "*Suggested New Knowledge Sharing Factors"* of Table, if required.

If there are any other suggestions, please mention in the column other Remarks.

Experts feedback form to record reviewers suggestions

Category	Knowledge Sharing Factors	Knowledge Sharing Sub-Factors	Description	Suggestions on Naminig Convention	Suggestions on Grouping/Sub-grouping	Suggested New Knowledge Sharing Factors/Sub-factors or Categories	Other Remarks
Individual	Individual perspective is most obvious lens in code review. The individual can be an author or reviewer						
	Individual Impartiality	It refers to the equal treatment of all individuals and the group.					
		Biasness	Biasness refers to attitude for or as opposed to one individual or group. For instance, reviewing code of selected authors.				
		Balance Between Equity and Equality	Equity refers to the distribution of resources in harmony with one's contribution. Equality is the state of being equal, particularly in position, privileges, that are all participants deserve the same resources, and irrespective of contribution.				

Appendix I Delphi Surveys Invitation Letter

19th June 2019

Invitation Letter

Dear Sir/Madam

Subject: Delphi Survey to develop a knowledge sharing model for modern code review to reduce software engineering waiting waste.

My name is Nargis Fatima. I am undertaking a Ph.D. research program with Razak Faculty of Technology and Informatics at Universiti Teknologi Malaysia (UTM). The title of my research is "Knowledge sharing model for modern code review to reduce software engineering waiting waste". I am writing to invite your participation in a Delphi survey. This study will help me in my PhD research in the development of knowledge sharing model for modern code review to reduce software engineering waiting waste. My Ph.D. research supervised by Associate Professor Dr. Suriayati Bt. Chuprat from Razak Faculty of Technology and Informatics at Universiti Teknologi Malaysia at UTM University.

The aim and objectives of conducting the Delphi survey are:

(a) To assess the recognized knowledge sharing factors, sub-factors, and categories for appropriate naming conventions, grouping, and sub-grouping.

(b) To assess the practicality of the recognized knowledge sharing factors, sub-factors, and categories in the context of MCR with industry to reduce software engineering waiting waste.

(c) To recognize the most influential knowledge sharing factors for MCR activities concerning the industry.

(d) To recognize new industry-based knowledge sharing factors, with their associated sub-factors and categories in the context of MCR for reducing software engineering waiting waste.

A panel of 10 experts will be surveyed using Delphi Technique. You have been selected and invited to contribute to this Delphi survey. Your knowledge,

experiences, and feedback will provide an invaluable contribution to my research. The data collected as part of the Delphi survey will seek to classify areas of consensus and disagreement among the panel members. The study is scheduled to be completed in one months. The summary result of each round will be made available to you for your reexamination in the next stage. Final report will be given to you at the conclusion of this research.

All panel members will maintain anonymity, each participant will be assigned a distinctive code which will only be known by the researcher and supervisor. We hope you are willing to contribute to the study. Thank you for your consideration and please do not hesitate to contact us for any inquiries.

Nargis Fatima
PhD Student
UTM
Kuala Lumpur, Malaysia
Email: fatimanargis@graduate.utm.my
Contact: +60102683914

Dr. Suriayati Chuprat
Assistant Professor
UTM
Kuala Lumpur, Malaysia
Email: suriayati.kl@utm.my

Appendix J Delphi Survey Questionnaires

The Questionaires for Round 1 and Round 2 were same except the Section I, in which the background information of the Delpji panel memebers was collected. Round 1 Delphi Round 1 questionnare has four sections and Round 2 has three sections.

Section I: We will begin with collecting some background information from you.

Section II: This section will request you to assess

1) The percieved level of practicality of knowledge sharing factors by assigning score to their associated sub factors. The score that should be assigned are distributed as (5=Very High, 4= High, 3= Moderate, 2 =Low, 1 Very Low)

2) The percived level of influence of listed knowledge sharing factors for MCR actvities by assigning the score to their associated sub-factors. The socre are distributed as (5=Most Influencial, 4= Influencial, 3= Moderate, 2 =Weakly influencial, 1 Not Infleuncial)

Section III: This section is designed to mention any new knowledge sharing factors, categories and associated sub factors.

Section IV: This section is designed to mention real project example for which the panel members were involved or had perfomed the MCR activities.

We have assigned you a user ID for this study and it is: _____

Please do not hesitate to contact the researcher at fatimanargis@graduate.utm.my. Once again, thank you for your time and your contribution to this research.

Regards,
Nargis Fatima
PhD Student
Razzak Faculty of Technology and Informatics
University Technology Malaysia

Questionaire Section I (Round I)

Background Information

Intsructions: In this section we would like to know about your background information Please tick in appropriate boxes or fill in the blanks.

Personal Information

First Name: _____ Last Name: _____

Company: _____

Email Address: _____

Phone Number: _____

How long you have been working in the field of _____

_____

How much of the time you spend in coding or code reviewing_____

_____

Number of projects approx you have been involved in code revewing _____

_____

Your experience in industry _____

Questionaire Section II for Round 1 and Section I for Round 2

Assement of knowledge sharing factors for their perceived level of practicality and perceieved level of influence

Instructions:

You need to assess the level of practicality of knowledge sharing factors by assigning score to their associated sub-factors. You also have to assess the percived level of influence of listed knowledge sjaring factors for each MCR actvity by assigning the score to their associated subfactors.

Scale to assess the practicallity of knowledge sharing factors.

1=Very High, 2= High, 3= Moderate, 4 =Low, 5 Very Low.

Scale for most influencial knowledge sharing factors.

1=Strongly Influencial, 2= Influencial, 3= Moderate, 4 =Weakly influencial, 5 Not Infleuncial

Form to record Delphi panel member feedback

Category	Knowledge Sharing Factors	Knowledge Sharing Su-factors	Description	Degree of Practicality	Knowleddge Sharing Factors influence for MCR actvity				
					SCP	SCS	RSN	SCR	SCA
Individual	Individual perspective is most obvious lens in code review. The individual can be an author or reviewer								
	Individual Impartiality	It refers to the equal treatment of all individuals and the group.							
		Biasness	It refers to attitude for or as against to one individual or group. For instance, reviewing code of particular authors.						
		Balance Between Equity and Equality	Equity refers to the distribution of treatment, resources, and outcomes in harmony with one's contribution Equality is the state of being equal, particularly in position, privileges, or opportunities that are all participants deserve the same treatments, irrespective of contribution.						
.	.	.	.						

Questionaire Section III for Round 1 and Section II for Round 2

Suggestion of new knowledge sharing factors, sub-factors, and categories

Template of form to enter suggestions concerning new knowledge sharing factors, subfactors, and categories

Category	Knowledge Sharing Factors	Knowledge Sharing Sub-factors	Description	Degree of Practicality	Knowleddge Sharing Factors Influence for MCR actvity				
					SCP	SCS	RSN	SCR	SCA

Questionaire Section IV for Round 1 and Section III for Round 2

Real Project Example

Please share any of your recent software project in which you were involved as a developer or reviewer.

Project Name

_____

Project Description _____

_____

Programming Language used_____

Thank you for your participation.

Regards,
Nargis Fatima
PhD Student
Razzak Faculty of Technology and Informatics,
University Technology Malaysia
fatimanargis@graduate.utm.my

Appendix K Delphi Survey Results

Mean, standard deviation, coefficient of variation for percieved level of practicality Round 1

Knowledge Sharing Factors (KSF)	Knowledge Sharing Sub-Factor (KSSbF)	MPPV (KSSbF) R1	σ(KSSbF) R1	CMPPV (KSF) R1	σ(KSF) R1	CV(KSF) R1
Individual Impartiality	Biasness	4.8	0.421637	4.65	0.58214164	0.125191751
	Balance Between Equity and Equality	4.5	0.707107			
Individual Historical Factors	Individual Characteristics	4.4	0.699206	4.575	0.450308536	0.098428095
	Individual Knowledge	5	0			
	Individual Experience	5	0			
	Individual Expertise	5	0			
	Individual Skills	4.3	0.483046			
	Work Style	4.2	0.421637			
	Work Track Record	4.3	0.483046			
	Affiliation	4.4	0.699206			
Individual Emotions	Feelings	4.4	0.516398	4.35	0.5	0.114942529
	Fear	4.3	0.483046			
Individual Pressure	Cognitive Load	4.8	0.421637	4.675	0.337474279	0.072187012
	Individual Workload	5	0			
	Time Pressure	4.1	0.316228			
	Context Switching	4.8	0.421637			
Individual Awareness	Awareness of Code Quality	4.5	0.527046	4.02	1.048808848	0.260897723
	Awareness of Process Improvement	3.7	0.823273			
	Awareness of Knowledge Sharing	3.9	1.197219			
	Awareness of effective Communication	4	1.054093			
	Awareness of Role-Oriented Tasks	4	1.414214			

Individual Turnover	Interpersonal Conflicts	4	1.054093	4	1.10888662	0.277221655
	Individual Matters	4.2	1.032796			
	Impact of Turnover	3.8	1.229273			
Individual Intentions	Self-Learning	5	0	4.52	0.382970843	0.084728063
	Collaboration	4.2	0.421637			
	Problem Solving	4.8	0.421637			
	Impression Formation	4.1	0.316228			
	Build Relationships	4.5	0.527046			
Relational	Trust	4.7	0.483046	4.2	0.532290647	0.126735868
	Reputation	4.2	0.421637			
	Familiarity	4.1	0.567646			
	Frequency of Interaction	3.8	0.632456			
Structural	Social Network	5	0	4.38	0.419435246	0.095761472
	Social Network Ties	4.3	0.483046			
	Network Channel	3.8	0.421637			
	Network Stability	4.2	0.421637			
	Social Network structure	4.3	0.483046			
	Social Political Structure	4.7	0.483046			
Source Code	Source Code Structure	5	0	4.88	0.187197049	0.038360051
	Source Code Complexity	5	0			
	Source Code Readability	5	0			
	Source Code Efficiency	4.8	0.421637			
	Source Code Associated Risks	5	0			
	Exception Handling	5	0			
	Adherence to Coding Standards	4	0			
	Source Code Change Motivation	5	0			
	Source Code Change Documentation	5	0			

	Source Code Change Scope	5	0			
	Nature of Change	4.9	0.316228			
	Change Impact	5	0			
	Source Code Change Revertability	4.8	0.421637			
Feedback	Feedback Language	5	0	4.635	0.463766498	0.100057497
	Feedback Temporal Aspects	4.5	0.527046			
	Feedback Targeted Object	4.8	0.421637			
	Feedback Usefulness	4.8	0.421637			
	Feedback Source	4.5	0.527046			
	Feedback Structure	4.7	0.483046			
	Feedback Training	4.7	0.483046			
	Feedback Size	4.8	0.421637			
	Feedback Cycle	3.7	0.483046			
	Feedback Content	4.8	0.421637			
	Feedback Perception	4.8	0.421637			
	Feedback Communication	4.8	0.421637			
	Feedback Frequency	4.3	0.674949			
	Defect Details Conveyed in Feedback	4.7	0.483046			
Testing	Test Results	4.8	0.421637	4.27	0.574133808	0.134457566
	Manual Tests	3.6	1.074968			
	Test Suits	4.7	0.483046			
	Test Quality	4.1	0.567646			
	Automated Tests	3.8	0.421637			
	Test Documentation	4.8	0.421637			
	Test Coverage	4.4	0.516398			
	Test Type	3.5	0.527046			
	Proof of Testing	4.8	0.421637			

Process Support	Development Process	4.4	0.516398	4.16	0.509175077	0.122397855
	Review Process	4.1	0.316228			
	Process Complexity	4.7	0.483046			
	Process Selection	3.8	0.632456			
	Process Quality	4.5	0.527046			
	Process Availability	3.5	0.527046			
Tool Support	Development Tool	4	0	4.66	0.347610894	0.074594612
	Review Tool	4.7	0.483046			
	Technical Maturity	4.7	0.483046			
	Integration of Review Tool with Development Tool	5	0			
	Automated Feature Assistance	5	0			
	Selection of Tool	4.3	0.483046			
	Tool Quality	5	0			
	Tool Availability	4.6	0.516398			
Organization Support	Availability of Resources	4.3	0.674949	4.24	0.46547466	0.109781759
	Organization Policies	4	0			
	Organization Characteristics	4.2	0.421637			
	Organizational Practices	4.3	0.483046			
Communication Support	Communication Type	5	0	4.86	0.316227766	0.065067442
	Communication Channel	5	0			
	Communication Purpose	4.6	0.516398			
	Communication Pattern	5	0			
	Communication Procedure	4.7	0.483046			
Project Support	Problem Domain	4.5	0.527046	4.46	0.519971509	0.11658554
	Project Quality Assessment	4.4	0.516398			
	Project Attributes	4.6	0.516398			
	Release Management	4.4	0.516398			

	Adherence to Standards	4.5	0.527046			
	Risk Management	4.4	0.516398			
Team Organization	Team Size	3.6	0.516398	4.24	0.45704364	0.107793311
	Team Roles	4.4	0.516398			
	Team Responsibilities	4.5	0.527046			
	Team Distance	5	0			
	Role Multiplicity	3.7	0.483046			
Team Strategies	Team Policies	4.7	0.483046	4.375	0.320589734	0.073277653
	Team Work Practices	5	0			
	Team Rules	4	0			
	Team Work Processes	3.8	0.421637			
Team Culture	Familiarity among Team Members	4.5	0.527046	4.55	0.521749195	0.114670153
	Friction among Team Members	4.6	0.516398			
	Team Accountability	4.5	0.527046			
	Team Values	4.6	0.516398			
Team Intentions	Identify Better Solutions	4.2	0.421637	4.644	0.281091348	0.060527853
	Improve Code Quality	5	0			
	Knowledge Distribution	5	0			
	Improve Development Process	5	0			
	Avoid Breaking Builds	4.2	0.421637			
	Share Code Ownership	4.2	0.421637			
	Increase Team Awareness	4.2	0.421637			
	Improve Software Quality	5	0			
	Identify Defects	5	0			
Team Drives	Team Productivity	4.3	0.483046	4.55	0.5	0.10989011
	Team Motivations	4.6	0.516398			
	Team Priorities	4.6	0.516398			

	Team Workload	4.8	0.421637			
	Team Cohesion	4.5	0.527046			
	Team Participation	4.5	0.527046			

Mean, standard deviation, coefficient of variation for percieved level of practicality Round 2

Knowledge Sharing Factors (KSF)	Knowledge Sharing Sub-Factor (KSSbF)	MPPV (KSSbF) R2	σ(KSSbF) R2	CMPPV (KSF) R2	σ(KSF) R2	CV(KSF) R2
Individual Impartiality	Biasness	4.7	0.483046	4.7	0.483045892	0.102776
	Balance Between Equity and Equality	4.7	0.483046			
Individual Historical Aspects	Individual Characteristics	4.6	0.516398	4.825	0.353553391	0.073275
	Individual Knowledge	5	0			
	Individual Experience	5	0			
	Individual Expertise	5	0			
	Individual Skills	5	0			
	Work Style	4.7	0.483046			
	Work Track Record	4.7	0.483046			
	Affiliation	4.6	0.516398			
Individual Emotions	Feelings	4.7	0.483046	4.5	0.483045892	0.107344
	Fear	4.3	0.483046			
Individual Load	Cognitive Load	5	0	4.725	0.263523138	0.055772
	Individual Workload	5	0			
	Time Pressure	4.1	0.316228			
	Context Switching	4.8	0.421637			
	Awareness of Code Quality	4.6	0.516398	4.14	0.880656321	0.212719

Individual Awareness	Awareness of Process Improvement	3.7	0.823273			
	Awareness of Knowledge Sharing	4	1.054093			
	Awareness of effective Communication	4	1.054093			
	Awareness of Role-Oriented Tasks	4.4	0.843274			
Individual Turnover	Interpersonal Conflicts	4.4	0.516398	4.333333	0.779363463	0.179853
	Individual Matters	4.4	0.966092			
	Impact of Turnover	4.2	0.788811			
Individual Intentions	Self-Learning	5	0	4.64	0.377123617	0.081277
	Collaboration	4.5	0.527046			
	Problem Solving	4.9	0.316228			
	Impression Formation	4.1	0.316228			
	Build Relationships	4.7	0.483046			
Social Relational Aspects	Trust	5	0	4.475	0.411636301	0.091986
	Reputation	4.8	0.421637			
	Familiarity	4.2	0.421637			
	Frequency of Interaction	3.9	0.567646			
Social Structural Aspects	Social Network	5	0	4.416667	0.382486988	0.086601
	Social Network Ties	4.3	0.483046			
	Network Channel	4	0			
	Network Stability	4.2	0.421637			
	Social Network structure	4.3	0.483046			
	Social Political Structure	4.7	0.483046			
Source Code	Source Code Structure	5	0	4.9	0.146176337	0.029832
	Source Code Complexity	5	0			
	Source Code Readability	5	0			
	Source Code Efficiency	4.8	0.421637			

	Source Code Associated Risks	5	0			
	Exception Handling	5	0			
	Adherence to Coding Standards	4	0			
	Source Code Change Motivation	5	0			
	Source Code Change Documentation	5	0			
	Source Code Change Scope	5	0			
	Nature of Change	5	0			
	Change Impact	5	0			
	Source Code Change Revertability	4.9	0.316228			
Feedback	Feedback Language	5	0	4.685714	0.423515147	0.090384
	Feedback Temporal Aspects	4.6	0.516398			
	Feedback Targeted Object	4.8	0.421637			
	Feedback Usefulness	5	0			
	Feedback Source	4.5	0.527046			
	Feedback Structure	4.7	0.483046			
	Feedback Training	4.7	0.483046			
	Feedback Size	4.8	0.421637			
	Feedback Cycle	3.7	0.483046			
	Feedback Content	4.8	0.421637			
	Feedback Perception	4.9	0.316228			
	Feedback Communication	4.8	0.421637			
	Feedback Frequency	4.3	0.674949			
	Defect Details Conveyed in Feedback	5	0			
Test Deliverables	Test Results	5	0	4.411111	0.443053379	0.10044
	Manual Tests	3.9	0.875595			
	Test Suits	4.9	0.316228			
	Test Quality	4.2	0.421637			

	Automated Tests	3.8	0.421637			
	Test Documentation	5	0			
	Test Coverage	4.4	0.516398			
	Test Type	3.5	0.527046			
	Proof of Testing	5	0			
Process Support	Development Process	4.4	0.516398	4.383333	0.436738756	0.099636
	Review Process	4.5	0.527046			
	Process Complexity	4.9	0.316228			
	Process Selection	4	0.471405			
	Process Quality	4.5	0.527046			
	Process Availability	4	0			
Tool Support	Development Tool	4	0	4.78	0.253859104	0.053109
	Review Tool	4.8	0.421637			
	Testing Tool	5	0			
	Technical Maturity	4.7	0.483046			
	Integration of Review Tool with Development Tool	5	0			
	Integration of Testing Tool with Development Tool	5	0			
	Automated Feature Assistance	5	0			
	Selection of Tool	4.3	0.483046			
	Tool Quality	5	0			
	Tool Availability	5	0			
Organization Support	Availability of Resources	4.4	0.516398	4.25	0.421637021	0.099209
	Organization Policies	4	0			
	Organization Characteristics	4.2	0.421637			
	Organizational Practices	4.4	0.516398			
Communication Support	Communication Type	5	0	4.88	0.298142397	0.061095
	Communication Channel	5	0			

	Communication Purpose	4.6	0.516398			
	Communication Pattern	5	0			
	Communication Procedure	4.8	0.421637			
Project Support	Problem Domain	4.7	0.483046	4.533333	0.512799145	0.113117
	Project Quality Assessment	4.4	0.516398			
	Project Attributes	4.6	0.516398			
	Release Management	4.6	0.516398			
	Adherence to Standards	4.5	0.527046			
	Risk Management	4.4	0.516398			
Team Organization	Team Size	3.6	0.516398	4.3	0.402768199	0.093667
	Team Roles	4.4	0.516398			
	Team Responsibilities	4.5	0.527046			
	Team Distance	5	0			
	Role Multiplicity	4	0			
Team Strategies	Team Policies	4.7	0.483046	4.425	0.241522946	0.054581
	Team Work Practices	5	0			
	Team Rules	4	0			
	Team Work Processes	4	0			
Team Culture	Familiarity among Team Members	4.7	0.483046	4.6	0.510990324	0.111085
	Friction among Team Members	4.6	0.516398			
	Team Accountability	4.5	0.527046			
	Team Values	4.6	0.516398			
Team Intentions	Identify Better Solutions	4.9	0.316228	4.722222	0.265274142	0.056176
	Improve Code Quality	5	0			
	Knowledge Distribution	5	0			
	Improve Development Process	5	0			
	Avoid Breaking Builds	4.2	0.421637			

	Share Code Ownership	4.2	0.421637			
	Increase Team Awareness	4.2	0.421637			
	Improve Software Quality	5	0			
	Identify Defects	5	0			
Team Drives	Team Productivity	4.2	0.421637	4.714286	0.402373908	0.085357
	Team Motivations	4.8	0.421637			
	Team Priorities	4.6	0.516398			
	Team Workload	5	0			
	Team Shared Vision	4.9	0.316228			
	Team Cohesion	4.7	0.483046			
	Team Participation	4.8	0.421637			

Difference of coefficient of variation among Round 1 and Round 2 for perceieved practicality

Knowledge Sharing Factors Round 1	**CV- Round 1**	**CV-Round 2**	**CV (R1-R2)**
Individual Impartiality	0.125191751	0.102775722	0.022416029
Individual Historical Aspects	0.098428095	0.073275314	0.025152781
Individual Emotions	0.114942529	0.107343531	0.007598997
Individual Load	0.072187012	0.055772093	0.016414919
Individual Awareness	0.260897723	0.212718918	0.048178805
Individual Turnover	0.277221655	0.179853107	0.097368548
Individual Intentions	0.084728063	0.081276642	0.003451421
Social Relational Aspects	0.126735868	0.091985766	0.034750103
Social Structural Aspects	0.095761472	0.086600828	0.009160644
Source Code	0.038360051	0.029831905	0.008528146
Feedback	0.100057497	0.09038433	0.009673167
Test Deliverables	0.134457566	0.100440313	0.034017254
Process Support	0.122397855	0.099636218	0.022761637
Tool Support	0.074594612	0.053108599	0.021486013
Organization Support	0.109781759	0.099208711	0.010573049
Communication Support	0.065067442	0.061094753	0.003972688
Project Support	0.11658554	0.113117458	0.003468082
Team Organization	0.107793311	0.093667023	0.014126288
Team Strategies	0.073277653	0.054581457	0.018696197
Team Culture	0.114670153	0.111084853	0.0035853
Team Intentions	0.060527853	0.056175701	0.004352152
Team Drives	0.10989011	0.085357214	0.024532896

Mean, standard deviation, coefficient of variation for percieved level of influence for MCR actvities Round 1

Knowledge Sharing Factors Round 1	Mean, Standard Deviation, Coefficient of Variation for Percieved Level of Influence for MCR Actvities Round 1								
	Source Code Preparation			Source Code Preparation			Source Code Preparation		
	MPIV (KSF)	σ(KSF)	CV (KSF)	MPIV (KSF)	σ(KSF)	CV (KSF)	MPIV (KSF)	σ(KSF)	CV (KSF)
Individual Impartiality	1.55	0.5	0.322581	1.5	0.516398	0.344265	4.6	0.516398	0.11226
Individual Historical Factors	4.5125	0.381881	0.084627	3.925	0.401386	0.102264	4.862	0.314024	0.064587
Individual Emotions	3.7	0.483046	0.130553	1.75	0.428174	0.244671	2.3	0.434613	0.188962
Individual Pressure	4.275	0.431406	0.100914	2.65	0.718795	0.271244	2.75	0.401386	0.145959
Individual Awareness	3.64	1.20185	0.330179	2.48	0.635959	0.256435	3.24	0.904311	0.279108
Individual Turnover	3.633333	1.08696	0.299163	2.633	0.490653	0.186348	3.833333	1.01653	0.265182
Individual Intentions	3.98	0.637704	0.160227	2.64	0.489898	0.185567	3.28	0.298142	0.090897
Relational	2.15	0.471405	0.219258	1.52	0.512076	0.336892	4.65	0.428174	0.092081
Structural	2	0.370185	0.185093	1.666667	0.471405	0.282843	4.783333	0.419435	0.087687
Source Code	4.915385	0.191708	0.039002	4.438	0.428673	0.096592	4.115	0.377803	0.091811
Feedback	4	0.394405	0.098601	1.742857	0.4291	0.246205	3.114	0.311168	0.099925
Testing	4.033333	0.356596	0.088412	4.4	0.486864	0.110651	1.68	0.424555	0.252711
Process Support	4.016667	0.419435	0.104424	4.183333	0.362604	0.086678	4.016667	0.481125	0.119782
Tool Support	4.5375	0.466964	0.102912	4.55	0.349603	0.076836	4.4875	0.457954	0.102051
Organization Support	4.325	0.397911	0.092003	4.175	0.337474	0.080832	4.275	0.345607	0.080844
Communication Support	1.84	0.382971	0.208136	2	0.34641	0.173205	1.84	0.426875	0.231997
Project Support	4.216667	0.507353	0.120321	4.15	0.319142	0.076902	4.11	0.215166	0.052352
Team Organization	4.4	0.405518	0.092163	2.74	0.418994	0.152917	3.54	0.469042	0.132498
Team Strategies	4.475	0.508265	0.113579	4.325	0.616892	0.142634	4.325	0.508265	0.117518
Team Culture	3.325	0.474342	0.142659	2.325	0.411636	0.177048	4.275	0.462481	0.108183
Team Intentions	3.311111	0.45542	0.137543	2.177778	0.28545	0.131074	3.722222	0.366835	0.098553
Team Drives	4.316667	0.377614	0.087478	2.183333	0.313286	0.14349	3.183333	0.401386	0.12609

Mean, standard deviation, coefficient of variation for percieved level of influence for MCR actvities Round 1

Knowledge Sharing Factors	Mean, Standard Deviation, Coefficient of Variation for Percieved Level of Influence for MCR Actvities Round 1					
	Source Code Review			Source Code Approval		
	MPIV (KSF)	σ(KSF)	CV (KSF)	MPIV (KSF)	σ(KSF)	CV (KSF)
Individual Impartiality	4.55	0.5	0.10989	3.85	0.582142	0.151206
Individual Historical Factors	4.275	0.316228	0.073971	4.725	0.376386	0.079658
Individual Emotions	4.15	0.372678	0.089802	2.1	0.316228	0.150585
Individual Pressure	4.775	0.390868	0.081857	3.225	0.491596	0.152433
Individual Awareness	3.34	0.971825	0.290966	3.5	1.210142	0.345755
Individual Turnover	2.7	0.498145	0.184498	2.3	0.451335	0.196233
Individual Intentions	4.62	0.368179	0.079692	2.72	0.485341	0.178434
Relational	3.37	0.485913	0.144188	3.225	0.390868	0.121199
Structural	3.566667	0.498145	0.139667	2.9	0.370185	0.12765
Source Code	4.915385	0.281935	0.057358	4.915385	0.275805	0.056111
Feedback	4.635714	0.283123	0.061074	2.185	0.287297	0.131486
Testing	4.788889	0.349603	0.073003	2.222222	0.41574	0.187083
Process Support	4.383333	0.419435	0.095689	4.383333	0.396746	0.090512
Tool Support	4.6375	0.281366	0.060672	4.45	0.361325	0.081197
Organization Support	4.125	0.456435	0.110651	4.375	0.468449	0.107074
Communication Support	4.7	0.469042	0.099796	3	0.442217	0.147406
Project Support	4.183333	0.352241	0.084201	4.266667	0.370185	0.086762
Team Organization	3.82	0.439697	0.115104	3.18	0.478423	0.150448
Team Strategies	3.85	0.372678	0.096799	4.55	0.521749	0.11467
Team Culture	2.95	0.324893	0.110133	4.15	0.477261	0.115003
Team Intentions	4.488889	0.272166	0.060631	2.2	0.412759	0.187618
Team Drives	3.983333	0.352241	0.088429	3.583333	0.6101	0.170261

Mean, standard deviation, coefficient of variation for percieved level of influence for MCR actvities Round 2

Knowledge Sharing Factors (KSF)	Mean, Standard Deviation, Coefficient of Variation for percieved level of influence for MCR actvities Round 2								
	Source Code Preparation			Source Code Review			Reviewer Selection and Notification		
	MPIV (KSF)	σ(KSF)	CV (KSF)	MPIV (KSF)	σ(KSF)	CV (KSF)	MPIV (KSF)	σ(KSF)	CV (KSF)
Individual Impartiality	1.6	0.50552503	0.315953144	1.6	0.516397779	3.165026	4.7	0.471405	0.100299
Individual Historical Aspects	4.6	0.349602949	0.076000641	4	0.387298335	11.44155	4.8875	0.300463	0.061476
Individual Emotions	3.9	0.316227766	0.081084043	1.8	0.365148372	5.6921	2.45	0.372678	0.152113
Individual Load	4.375	0.337474279	0.077136978	2.675	0.513701167	7.92653	2.775	0.390868	0.140853
Individual Awareness	4	0.837987006	0.209496751	2.58	0.47375568	3.078807	3.36	0.676593	0.201367
Individual Turnover	3.83	0.879814795	0.229716657	2.7	0.451335467	3.068828	3.966667	0.818761	0.20641
Individual Intentions	4.06	0.418993503	0.10320037	2.68	0.452155332	6.396281	3.32	0.266667	0.080321
Social Relational Aspects	2.35	0.440958552	0.187641937	1.575	0.450308536	3.571764	4.775	0.411636	0.086207
Social Structural Aspects	2.033333333	0.327730693	0.16117903	1.7	0.455420034	5.187186	4.833333	0.360041	0.074491
Source Code	4.923076923	0.170469437	0.034626604	4.469230769	0.395487366	26.2172	4.130769	0.359249	0.086969
Feedback	4.1	0.333333333	0.081300813	1.757142857	0.414039336	5.271429	3.121429	0.307318	0.098454
Test Deliverables	4.088888889	0.293972368	0.071895416	4.45	0.472712164	15.13748	1.7	0.411261	0.241918
Process Support	4.066666667	0.380058475	0.093457002	4.25	0.275546595	11.18249	4.1	0.403687	0.09846
Tool Support	4.63	0.357460176	0.077205222	4.51	0.202758751	12.61679	4.52	0.194365	0.043001
Organization Support	4.4	0.357460176	0.081240949	4.1	0.298142397	11.46981	4.3	0.223607	0.052002
Communication Support	1.86	0.355902608	0.191345488	1.94	0.21602469	5.450929	1.88	0.382971	0.203708
Project Support	4.333333333	0.434613494	0.100295422	4.183333333	0.215165741	9.625411	4.15	0.129099	0.031108
Team Organization	4.44	0.359010987	0.08085833	2.76	0.394405319	7.687787	3.58	0.449691	0.125612
Team Strategies	4.575	0.437797518	0.095693447	4.4	0.45338235	10.05031	4.475	0.41833	0.093482
Team Culture	3.375	0.437797518	0.129717783	2.3	0.387298335	5.25357	4.4	0.453382	0.103041
Team Intentions	3.455555556	0.414252264	0.119880076	2.177777778	0.204878766	5.257129	3.755556	0.318174	0.084721
Team Drives	4.442857143	0.311167795	0.070037767	2.314	0.21821789	7.436502	3.342857	0.39841	0.119182

Mean, standard deviation, coefficient of variation for percieved level of influence for MCR actvities Round 2

Knowledge Sharing Factors	Mean, Standard Deviation, Coefficient of Variation for percieved level of influence for MCR actvities Round 2					
	Source Code Review			Source Code Approval		
	MPIV (KSF)	σ(KSF)	CV (KSF)	MPIV (KSF)	σ(KSF)	CV (KSF)
Individual Impartiality	4.65	0.5	0.107527	4	0.459468	0.114867
Individual Historical Aspects	4.3	0.278887	0.064857	4.75	0.353553	0.074432
Individual Emotions	4.35	0.341565	0.078521	2.1	0.298142	0.141973
Individual Load	4.925	0.263523	0.053507	3.275	0.491596	0.150106
Individual Awareness	3.64	0.656591	0.180382	3.56	0.845905	0.237614
Individual Turnover	3.033333	0.426006	0.140442	2.366667	0.434613	0.18364
Individual Intentions	4.7	0.270801	0.057617	2.76	0.476095	0.172498
Social Relational Aspects	3.6	0.440959	0.122488	3.3	0.387298	0.117363
Social Structural Aspects	3.633333	0.46746	0.128659	2.933333	0.327731	0.111726
Source Code	4.961538	0.191708	0.038639	4.884615	0.269536	0.055181
Feedback	4.685714	0.191071	0.040777	2.2	0.281718	0.128054
Test Deliverables	4.822222	0.302255	0.06268	2.222222	0.412759	0.185742
Process Support	4.43	0.370185	0.083563	4.5	0.360041	0.080009
Tool Support	4.7	0.266667	0.056738	4.6	0.274874	0.059755
Organization Support	4.175	0.383695	0.091903	4.4	0.453382	0.103041
Communication Support	4.76	0.416333	0.087465	3.04	0.405518	0.133394
Project Support	4.2	0.344265	0.081968	4.533333	0.327731	0.072294
Team Organization	3.84	0.416333	0.10842	3.22	0.459468	0.142692
Team Strategies	4.1	0.258199	0.062975	4.65	0.5	0.107527
Team Culture	3.075	0.241523	0.078544	4.2	0.414997	0.098809
Team Intentions	4.533333	0.210819	0.046504	2.311111	0.388094	0.167925
Team Drives	4	0.338062	0.084515	3.642857	0.536005	0.147139

Difference of coeeficient of variation among Round 1 and Round 2 for percieved level of influence for MCR activities

Knowledge Sharing Factors	Perceived Level of Influence of Knowledge Sharing Factors CV Round 1- Round2				
	Source Code Preparation	Source Code Submission	Reviewer Selection and Notification	Source Code Review	Source Code Approval
Individual Impartiality	0.006628	0.021517	0.011962	0.002363	0.036339
Individual Historical Aspects	0.008627	0.005439	0.003112	0.009114	0.005226
Individual Emotions	0.049469	0.041811	0.036849	0.011281	0.008612
Individual Load	0.023777	0.079206	0.005105	0.02835	0.002327
Individual Awareness	0.120682	0.072809	0.077741	0.110584	0.108141
Individual Turnover	0.069447	0.019186	0.058771	0.044056	0.012593
Individual Intentions	0.057027	0.016853	0.010576	0.022075	0.005936
Social Relational Aspects	0.031616	0.050982	0.005874	0.021699	0.003836
Social Structural Aspects	0.023914	0.014949	0.013196	0.011008	0.015924
Source Code	0.004375	0.0081	0.004842	0.018719	0.00093
Feedback	0.017301	0.010573	0.001471	0.020297	0.003432
Test Deliverables	0.016517	0.004424	0.010793	0.010323	0.001341
Process Support	0.010967	0.021844	0.021322	0.012125	0.010503
Tool Support	0.025707	0.031878	0.05905	0.003934	0.021441
Organization Support	0.010762	0.008115	0.028842	0.018748	0.004033
Communication Support	0.016791	0.061852	0.028289	0.012331	0.014012
Project Support	0.020026	0.025468	0.021243	0.002233	0.014469
Team Organization	0.011305	0.010017	0.006886	0.006684	0.007756
Team Strategies	0.017885	0.039593	0.024036	0.033824	0.007143
Team Culture	0.012941	0.008657	0.005141	0.031589	0.016194
Team Intentions	0.017663	0.036997	0.013832	0.014127	0.019692
Team Drives	0.01744	0.049186	0.006908	0.003913	0.023122

Appendix L Result of Regression Analysis

Relationship between knowledge sharing factors and sub-factors in terms of categories.

Relationship	p-value (sig. (2-tailed))	β
Facility Conditions -> Individual		
Project Support->Individual Turnover	0.014	-0.5
Communication Support-> Individual Emotions	0.000	3.7
Process Support->Individual Turn Over	0.001	-1.94
Project Support ->Individual Emotion	0.004	-0.7
Availability of Resources->Individual Load	0.00	-0.7
Facility Condition->Team		
Availability of Resources-> Team Drive	0.000	0 .249
Facility Condition ->Artefact		
Process Selection->Feedback	0.004	0.021
Availability of Resources->Test Deliverable	0.00	0.203
Individual ->Artefact		
Individual Historical Aspects->Source Code	0.003	0.29
Individual Load->Test Deliverable	0.018	-0.633
Individual Emotions->Feedback	0.09	0.237
Individual Intention ->Feedback	0.049	0.716
Biasness->Feedback	0.00	-0.06
Awareness of knowledge sharing->Feedback	0.00	0.157
Awareness of Code Quality->Feedback	0.00	0.140
Affiliation->Feedback	0.00	0.121
Adherence to Standards->Feedback	0.00	0.010
Awareness of Code Quality->Source Code	0.00	0.080
Individual->Team		
Individual Impartiality->Team Culture	0.00	-1.2
Individual Load->Team Culture	0.002	1.57
Individual Turnover->Team Culture	0.008	-0.41
Individual Intentions->Team Intentions	0.003	0.4
Individual->Social		
Individual Intentions->Social Structural Aspects	0.007	0.7
Individual Turnover->Social Structural Aspects	0.05	-0.165
Awareness of Code Quality->Social Relational Aspects	0.00	0.5
Affiliation->Social Structural Aspects	0.00	0.585
Artefact -> Individual		
Source Code->Individual Historical Aspects	0.009	-0.1
Test Deliverable->Individual Impartiality	0.036	-0.9
Source Code->Individual Emotions	0.04	-2.7
Adherence to Standards->Feedback	0.00	0.010
Automated Test->Individual Load	0.00	-0.250
Artefact ->Team		
Adherence to Standards->Team Drive	0.00	0.223
Team -> Individual		

Relationship	p-value (sig. (2-tailed))	β
Team Intentions->Individual Turnover	0.00	3.6
Team Organization->Individual Load	0.043	-0.4
Team Intentions->Individual Intentions	0.003	1.2
Team->Social		
Team Strategies->Social Structural Aspects	0.003	1.1
Team->Artefact		
Team Intentions->Test Deliverable	0.00	1.75
Team Strategies->Test Deliverable	0.008	0.6
Team Organization->Test Deliverable	0.25	0.39
Familiarity among team members->Feedback	0.00	-0.15
Social-> Individual		
Social Structural Aspects->Individual Turnover	0.001	-1.4
Social Relational Aspects->Individual Intentions	0.06	0.34
Social -> Team		
Social Structural Aspects->Team Strategies	0.029	0.4

Appendix M Experiment Material

Instructions for Subjects (Author)

1. Read the given problems statement carefully and write the source code for the given problem using C++.

2. After completing the source code submit the source code to the facilitator and wait for the feedback on the source code

3. After receiving the feedback, read and understand the feedback and make corrections on the source and resubmit the source code to the facilitator and wait for the feedback. You can exchange comments with the reviewer who have provided the feedback for any clarification.

4. Repeat the cycle until your source code is approved by the reviewer.

Instructions for Subjects (Reviewers)

1. Review the source code given to you.

2. Write your feedback and give that feedback to the reviewer and waits for the resubmission of source code by the subject (author). You can exchange comments with the reviewer who have provided the feedback for any clarification.

3. Repeat the cycle until source code is approved.

Problem Statement for Session I

Create a Tic Tac Toe game with its basic functionality that can be played by two players on single standard computer system. Use C++ programming language to program the Tic Tac Toe game.

Problem Statement Session II

Design a bank administration application using C++ programming language having following features.

The user can create database.
The user can add new record to the database.
The user can search customer by name.
The user can search customer by phone number.
The user can modify customer data by name.
The user can view your database.

LIST OF PUBLICATIONS

Indexed Journal

1. **Nargis. F.**, Sumaira. N., & Suriayati. C. (2020). "Knowledge Sharing Framework for Modern Code Review to Diminish Software Engineering Waste" *International Journal of Advanced Computer Science and Applications (IJACSA),* 11(6), http://dx.doi.org/10.14569/IJACSA.2020.0110656. (**Indexed by SCOPUS & WOS)**

2. **Nargis. F.**, Sumaira. N., & Suriayati. C. (2020). " Knowledge Sharing Factors for Modern Code Review to Minimize Software Engineering Waste" *International Journal of Advanced Computer Science and Applications,* 11(1), 490–497. http://dx.doi.org/10.14569/IJACSA.2020.0110160. (**Indexed by SCOPUS & WOS)**

3. Sumaira. N., **Nargis. F**., & Suriayati. C. (2020). "Situational Modern Code Review Framework to Support Individual Sustainability of Software Engineers", *International Journal of Advanced Computer Science and Applications (IJACSA),* 11(6), 2020. 366-375, doi: 10.14569/IJACSA.2020.0110648. **(Indexed by SCOPUS &WOS)**

4. Sumaira. N., **Nargis. F**., & Suriayati. C. (2020). "Situational Factors for Modern Code Review to Support Software Engineers' Sustainability" International Journal of Advanced Computer Science and Applications (IJACSA), 11(1), 2020. 498-504 http://dx.doi.org/10.14569/IJACSA.2020.0110161. **(Indexed by SCOPUS & WOS)**

5. Sumaira. N., **Nargis. F**., Suriayati. C., Sarkan. H.. F., & (2020), Sarkan, Haslina, F, Nurulhuda & Sjarif, Nilam. (2020) "Sustainable Software Engineering: A Perspective of Individual Sustainability", *International Journal on Advanced Science, Engineering and Information Technology*, doi: 10. 676. 10.18517/ijaseit.10.2.10190. **(Indexed by SCOPUS)**

Indexed Conference Proceedings

1. **Nargis. F**., Sumaira. N., & Suriayati. C. (2020). "Software engineering wastes-A perspective of modern code review. *ACM International Conference Proceeding Series,* 93–99. https://doi.org/10.1145/3378936.3378953. **(Indexed by SCOPUS)**

2. Sumaira. N., **Nargis. F**., & Suriayati. C. (2020). "Modern Code Review Benefits-Primary Findings of A Systematic Literature Review". *ACM International Conference Proceeding Series,* pp 210-215, doi: 10.1145/3378936.3378954. **(Indexed by SCOPUS)**

3. **Nargis. F.,** Sumaira. N., & Suriayati. C. (2019). "Understanding the Impact of Feedback on Knowledge Sharing in Modern Code Review". *6th IEEE International Conference on Engineering Technologies and Applied Sciences (ICETAS)*, Kuala Lumpur, Malaysia. 10.1109/ICETAS48360.2019.9117268. **(Indexed by SCOPUS)**

4. **Nargis. F**., Sumaira. N., & Suriayati. C. (2019). "Knowledge sharing, a key sustainable practice is on risk: An insight from Modern Code Review". *6th IEEE International Conference on Engineering Technologies and Applied Sciences (ICETAS)*, Kuala Lumpur, Malaysia, doi: 10.1109/ICETAS48360.2019.9117444. **(Indexed by SCOPUS)**

5. **Nargis. F**., Sumaira. N., & Suriayati. C. (2019). "Individual, Social and Personnel Factors Influencing Modern Code Review Process," *IEEE Conference on Open Systems (ICOS),* Pulau Pinang, Malaysia, pp. 40-45, doi: 10.1109/ICOS47562.2019.8975708. **(Indexed by SCOPUS)**

6. Sumaira. N., **Nargis. F**., & Suriayati. C. (2019) "Does Project Associated Situational Factors have Impact on Sustainability of Modern Code Review Workforce?," *IEEE 6th International Conference on Engineering Technologies and Applied Sciences (ICETAS)*, Kuala Lumpur, Malaysia, doi: 10.1109/ICETAS48360.2019.9117541. **(Indexed by SCOPUS).**

7. Sumaira. N., **Nargis. F**., & Suriayati. C. (2019), " Situational factors affecting Software Engineers Sustainability: A Vision of Modern Code Review," *IEEE 6th International Conference on Engineering Technologies and Applied Sciences (ICETAS),* Kuala Lumpur, Malaysia, doi: 10.1109/ICETAS48360.2019.9117366. **(Indexed by SCOPUS)**

8. Sumaira. N., **Nargis. F**., & Suriayati. C. (2019) "Individual Sustainability Barriers and Mitigation Strategies: Systematic Literature Review Protocol," *IEEE Conference on Open Systems (ICOS),* Pulau Pinang, Malaysia, doi: 10.1109/ICOS47562.2019.8975707. **(Indexed by SCOPUS)**

9. **Nargis. F**., Sumaira. N., & Suriayati. C. (2018). "Challenges and Benefits of Modern Code Review-Systematic Literature Review Protocol*"*. International Conference on Smart Computing and Electronic Enterprise (ICSCEE)*, Shah Alam,* doi: 1109/ICSCEE.2018.8538393. **(Indexed by SCOPUS)**