Quest Journals
Journal of Software Engineering and Simulation
Volume 7 ~ Issue 10 (2021) pp: 39-45
ISSN(Online) :2321-3795 ISSN (Print):2321-3809
www.questjournals.org

Research Paper

The Risks and Mitigations Software Development Projects in Indonesia

Setio Ardy Nuswantoro
Study Program
Computer Science
Faculty of Engineering and Informatics, Muhammadiyah University of Palangkaraya

Abstract*—Currently, in various fields have succeed in software implementation, but the success rate is still quite low. It was reported that the success rate of software development projects in Asia was only 22%, while in Indonesia the success rate of software implementation was only 50%. The purpose of this research is to identify the risk of software development failure in Indonesia. This study is designed to answer three questions: (A) What are the risks in a software development project, (B) is there any difference stages of software development for the risks found and (C) What are the strategies for mitigating the identified risks. Using the Delphi method with a series of repeated questionnaires aimed at obtaining panelists' agreement. Research using the Delphi method involving 41 panelists with different regions and roles in software development and obtained 17 risk of failure to reach panelists agreement. It was found that each development stage (planning, analysis, design, development, implementation and maintenance) turned out to have different dominant risks with the risk of needs analyst less or business process being the most dominant risk at the analysis and design stage. Most of the panelists raised analysis, evaluation and communication as risk mitigation strategies.*
***Keywords*—** *Mitigations, Projects, Software, Risk*

Received 06 October, 2021; Revised: 18 October, 2021; Accepted 20 October, 2021 © The author(s) 2021. Published with open access at www.questjournals.org

I. INTRODUCTION

The concept of wisdom which refers to how people make right use of their knowledge through their practical actions, judgments, and ethical decisions, in general attracts researcher interest in a variety of disciplines, such as philosophy, psychology and management studies, little is known about how wisdom is conceptualized and then operationalized in the software development project team context. Based on the frameworks for philosophical, group and organizational wisdom, this paper identifies software development project team wisdom as a process for how team members best use the stock and flow of their knowledge through collective judgment, virtue-ethics, emotions/feelings, and effective decision-making during their project-related efforts. Adapting the efforts and functional similarities of both group and organizational wisdom practices, this effort determines that wisdom-related mechanisms (e.g., team diversity, networking with other teams and people, and their past experiences), joint epistemic actions (e.g., team reasoning, intuition, and aesthetic capacity), and team virtue and prudence become the different faces of the software development team wisdom process. We then propose how these different faces interrelate and how they also relate to project process effectiveness, such as team learning and speed-to-users, both of which have been rarely addressed empirically in the context of software development project teamwork. By examining 210 in-house software development project teams in a field study and using structural equation modeling analysis, our results empirically show the following: (a) software development wisdom-related mechanisms positively relate to software development team prudence and virtue and their joint epistemic actions, (b) software development team prudence and virtue are positively associated with software development team joint epistemic actions, and further (d) software development team joint epistemic actions are positively associated with software development project process effectiveness.

Analysis and thought leadership are offered in the CHAOS Manifesto series of reports. Another major change is how we define success. We have multiple definitions, including our newest. We coded the new CHAOS database with six individual attributes of success: OnTime, OnBudget, OnTarget, OnGoal, Value, and

Satisfaction. Our Traditional definition is OnTime, OnBudget, and OnTarget. This means the project was resolved within a reasonable estimated time, stayed within budget, and contained a good number of the estimated features and functions. Our new Modern definition is OnTime, OnBudget, with a satisfactory result. This means the project was resolved within a reasonable estimated time, stayed within budget, and delivered customer and user satisfaction regardless of the original scope. We have the flexibility to present the results for one to six of these attributes in any combination. The Traditional resolution of all software projects from FY2011-2015 within the new CHAOS database. The percentage of projects that were OnBudget from FY2011-2015 within the new CHAOS database. The percentage of projects that were OnTime from FY2011-2015 within the new CHAOS database. The percentage of projects that were OnTarget from FY2011-2015 within the new CHAOS database [1].

Design/methodology/approach: The top ten reasons of process improvement projects termination or failure are based on literature, interaction of authors with Lean Six Sigma Master Black Belts, consultants, practitioners and trainers on various topics of Lean, Six Sigma, general quality management and continuous improvement along several years' experience of the authors. Findings: The top ten reasons in our opinion include lack of commitment and support from top management; poor communication practices; incompetent team; inadequate training and learning; faulty selection of process improvement methodology and its associated tools/techniques; inappropriate rewards and recognition system/culture; scope creepiness; sub-optimal team size and composition; inconsistent monitoring and control; and resistance to change. Research limitations/implications: The top ten reasons mentioned in this study are based on only literature and authors' opinion. The authors of this paper have been pursuing a global study to critically evaluate the reasons behind process improvement projects failure based on a case-study approach. Originality/value: The chief operations officers and senior executives of various businesses can use these top ten reasons to develop project failure risk mitigation strategies and save significant cash-savings associated with such project terminations or failures in some other cases [2].

Software development life cycle or SDLC for short is a methodology for designing, building, and maintaining information and industrial systems [3]. So far, there exist many SDLC models, one of which is the Waterfall model which comprises five phases to be completed sequentially in order to develop a software solution [4]. However, SDLC of software systems has always encountered problems and limitations that resulted in significant budget overruns, late or suspended deliveries, and dissatisfied clients. The major reason for these deficiencies is that project directors are not wisely assigning the required number of workers and resources on the various activities of the SDLC [5]. Consequently, some SDLC phases with insufficient resources may be delayed; while, others with excess resources may be idled, leading to a bottleneck between the arrival and delivery of projects and to a failure in delivering an operational product on time and within budget. This paper proposes a simulation model for the Waterfall development process using the Simphony.NET simulation tool whose role is to assist project managers in determining how to achieve the maximum productivity with the minimum number of expenses, workers, and hours. It helps maximizing the utilization of development processes by keeping all employees and resources busy all the time to keep pace with the arrival of projects and to decrease waste and idle time. As future work, other SDLC models such as spiral and incremental are to be simulated, giving project executives the choice to use a diversity of software development methodologies.

II. METHOD

The data collection method used in this study is the Delphi method by making preparations to obtain agreement from the panelists [6], [7]. The question round in the Delphi method requires at least two to three rounds of questions to get agreement based on controlled feedback from the panelists [8].

The round of questions can be stopped if the answers of the panelists have reached an agreement, and if in the third round there is no agreement, the round of questions can be added until an agreement is obtained between the panelists [8].

The use of the Delphi method can provide results that can be said to be better than traditional survey methods [9].

Since the infancy of the Delphi Technique for collecting and aggregating expert insight, this methodological tool has been discussed, adapted and applied in over 2,600 published scholarly papers to date. This paper mines the major citation indexing services to analyze five dimensions of these data: primary contribution (methodological or applied), field and subfield, length (in pages), year, and journal/conference. Interpreted visual analytics of these five dimensions (both individually and in combination) provide researchers, practitioners and editors with clear insights about whether the Delphi technique is still as prominently used, discussed, and written about in the academic literature as it was twenty years ago and the related trends that might inform predictions of its future use. Among these insights, a simple time series of frequencies of Delphi publications by year immediately shows that academic acceptance of Delphi as a research tool is not only well

established, but it has been growing in popularity and range of research domains for two decades predicting unprecedented levels of use in the years to come.

Our first objective was to create empirically generated lists of risk factors for both domestically- and offshore-outsourced projects. Our second objective was to compare these two contexts: how do the risk factors change and which ones are most important in each. To address these objectives, we conducted two Delphi surveys to identify the important risk factors from a client perspective, in domestic and offshore settings. We qualitatively compared the results of the surveys to identify similarities and differences across their risk profiles. We identified three types of risks: those that appeared in both contexts; those that appeared in both but were exacerbated in the offshore context; and those that were unique to the offshore context. Our findings suggested that traditional project management risks were important in both contexts; however, the offshore context seemed to be more vulnerable to some traditional risks as well as factors that were unique to it.

III. RESULTS AND DISCUSSION

In the first stage of data collection, the authors distributed open-ended questions to 41 panelists to provide their views regarding the 5 risks that are considered to have the most impact on the failure of software development projects based on the experience of each panelist.

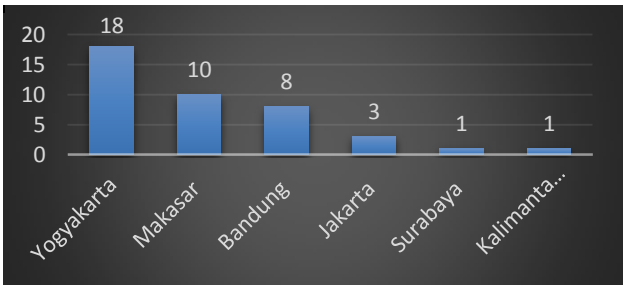

Fig. 1. Location of the First Stage Panelists

The location or region of the panelists who contributed to the first stage of data collection by providing their responses. In this first stage of data collection, Yogyakarta is the city that dominates the location of the panelists with a total of 18 panelists.

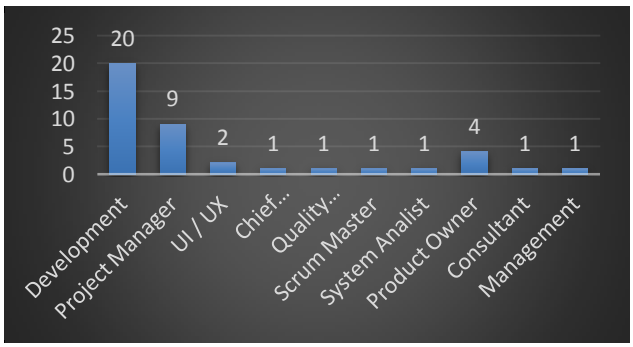

Fig. 2. The Role of the First Stage Panelists

In the third stage of data collection, the authors distributed questions to 32 panelists and 27 panelists contributed to this round of stages.

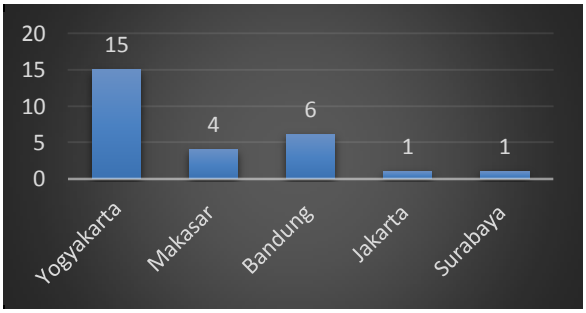

Fig. 3. Location of the Third Stage Panelists

The location of the panelists working on a software development project which again participated in the third stage, where the city of Yogyakarta still dominated the panelist location with a total of 15 panelists.

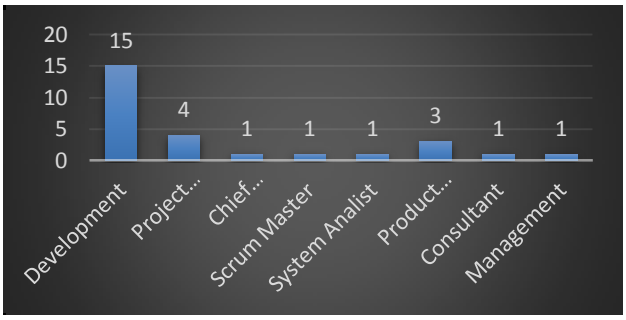

Fig. 4. Third Stage Panelist Role

The role of each panelist in the software development project who returned to participate in the third stage, the panelist who played the role of development still dominated his participation with a total of 15 panelists.

In the fourth stage of data collection, the authors distributed questions to 27 panelists, 14 panelists were willing to give their responses and as many as 13 panelists chose to withdraw as participants in the Delphi survey.

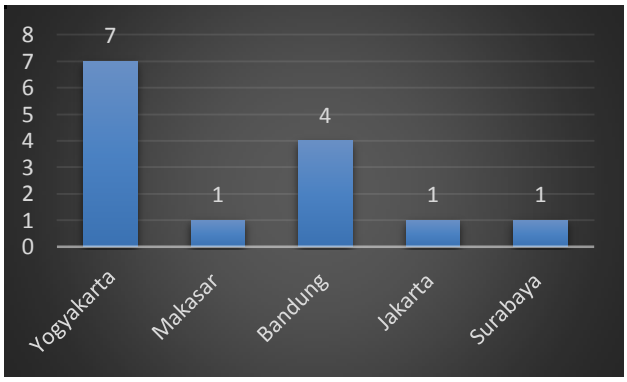

Fig. 5. Location of the Fourth Stage Panelists

The location or region of the panelists who contributed to the fourth stage of data collection. In this fourth stage of data collection, Yogyakarta still dominates the location of the panelists with a total of 7 panelists.

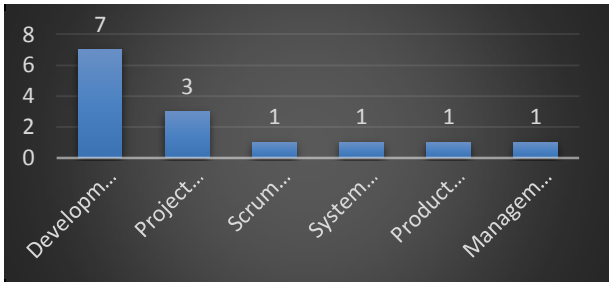

Fig. 6. The Role of the Fourth Stage Panelist

The role of each panelist in the software development project involved in the fourth stage. At this stage the development role still dominates with a total of 7 panelists with a development role.

TABLE I. FIRST STAGE RISK RESULT

Code	Failure Risk	Number of Responses
R.01	Lack of needs analysis/business process	21
R.02	Incorrect time estimate	19
R.03	Cost estimates do not match	18
R.04	Lack of communication between teams/users	18

	and developers	
R.05	System planning is not clear	17
R.06	Lack of expertise and knowledge of the development team	16
R.07	Unprofessional team	12
R.08	Lack of development team	11
R.09	Failure to get user commit	9
R.10	Requirements are not clear	9
R.11	Project management is not clear	8
R.12	Lack of user understanding	7
R.13	Inadequate facilities	7
R.14	Lack of user trust	6
R.15	Incorrect target market	5
R.16	Scope does not match	5
R.17	Lack of teamwork	4
R.18	Poor system monitoring	4
R.19	Low user acquisition	3
R.20	Cultural differences	2
R.21	Untested technology	2
R.22	Lack of data	2
Total		**205**

That Lack of needs analysis / business process occupied the risk most mentioned by the panelists. Meanwhile, cultural differences, untested technology and lack of data were the risks that were rarely mentioned by the panelists. Although cultural differences are rarely mentioned, it is proven that cultural differences are one of the risks that have an impact on the failure of software development projects.

The second stage of data collection aims to obtain a consensus based on the results obtained from the first stage [10]. The consensus assessment uses a Likert scale of 1-5 starting from the risk of less impact to the risk of having a very high impact on the failure of the software project. Consensus assessment is based on the point of view of each panelist role.

TABLE II. RISK BY DEVELOPMENT STAGE

Code	Failure Risk	Software Development Stages					
		Planning	Analyze	Design	Develop	Implement	Maintain
R.01	Lack of needs analysis/business process	27,14%	37,14%	20,00%	8,57%	5,71%	1,43%
R.02	Incorrect time estimate	29,41%	25,00%	16,18%	17,65%	10,29%	1,47%
R.04	Lack of communication between teams/users and developers	15,07%	17,81%	19,18%	27,40%	16,44%	4,11%
R.05	System planning is not clear	35,19%	31,48%	14,81%	7,41%	7,41%	3,70%
R.06	Lack of expertise and knowledge of the development team	15,94%	15,94%	17,39%	28,99%	14,49%	7,25%
R.07	Unprofessional team	18,60%	16,28%	17,44%	23,26%	15,12%	9,30%
R.08	Lack of development team	19,61%	17,65%	11,76%	29,41%	13,73%	7,84%
R.11	Project management is not clear	30,88%	20,59%	13,24%	17,65%	10,29%	7,35%
R.12	Lack of user understanding	19,64%	26,79%	16,07%	5,36%	19,64%	12,50%
R.13	Inadequate facilities	22,22%	11,11%	12,70%	23,81%	22,22%	7,94%
R.14	Lack of user trust	23,73%	13,56%	10,17%	15,25%	16,95%	20,34%
R.16	Scope does not match	25,81%	32,26%	16,13%	14,52%	8,06%	3,23%
R.17	Lack of teamwork	15,29%	15,29%	17,65%	23,53%	16,47%	11,76%
R.18	Poor system monitoring	14,29%	17,46%	12,70%	14,29%	20,63%	20,63%
R.19	Low user acquisition	14,89%	12,77%	17,02%	4,26%	27,66%	23,40%
R.20	Cultural differences	19,40%	22,39%	14,93%	11,94%	19,40%	11,94%
R.21	Untested technology	17,86%	16,07%	8,93%	17,86%	23,21%	16,07%

In the group and individual consensus assessments it was found that there was a difference in risk. Consensus Development Assessment obtained 20 risks of failure to reach consensus. Product Owner Consensus Assessment obtained 10 risks of failure to reach consensus. Project Manage Consensus Assessment obtained 21 risks of failure that reached consensus. While the consensus assessment calculated from the whole panelist or group obtained 17 risks that reached consensus among the panelists.

Of the 22 risks measured using SD and IR, 5 risks did not reach consensus. This is interesting because the 5 risks have a high number of responses and mean values. Such as cost estimation does not match 18 responses and mean value 4.06, failure to get user commitment with 9 responses and mean value 3.96, unclear

requirements with 9 responses and mean value 3.90, inappropriate target market with 5 responses and the mean value is 3.93 and the last is the lack of precise data with 2 responses and the mean value is 3.87. The 17 risks that reached group consensus were then ranked based on the highest average value.

TABLE III. LIST OF FAILURE RISKS

Code	Failure Risk	Mean	Rank
R.05	System planning is not clear	4,68	1
R.01	Lack of needs analysis/business process	4,65	2
R.07	Unprofessional team	4,56	3
R.04	Lack of communication between teams/users and developers	4,34	4
R.17	Lack of teamwork	4,31	5
R.11	Project management is not clear	4,28	6
R.06	Lack of expertise and knowledge of the development team	4,25	7
R.02	Incorrect time estimate	4,21	8
R.16	Scope does not match	3,93	9
R.18	Poor system monitoring	3,93	9
R.08	Lack of development team	3,84	10
R.13	Inadequate facilities	3,68	11
R.14	Lack of user trust	3,59	12
R.12	Scope does not match	3,40	13
R.21	Untested technology	3,34	14
R.19	Low user acquisition	3,12	15
R.20	Cultural differences	2,81	16

Judging from the average value of failure risk as shown in the table, it was found that each risk did not have a range of values that was too far away, such as the risk of unclear system planning and the risk of lack of analysis of business needs/processes which only had a value range of 0.03. This indicates that these risks have a similar impact on the failure of software development projects. In fact, there are also risks with the same average value, namely the risk of inappropriate scope and the risk of poor system monitoring with an average value of 3.93. The risk with the farthest value range is the risk of low user acquisition and the risk of cultural differences with a range of values reaching 0.31. Although cultural differences have the farthest value range and the lowest average value, cultural differences are still one of the risks that have an impact on the failure of software development projects.

During project development, the implementation of a risk management strategy is required by linking risk assessment and mitigation to assist in identifying risks and mitigating risks. In this study, the authors visualized the risk based on the most dominant risk [11]. The risk visualization was obtained from the results of the third round of questions. Panelists who are involved in this stage are asked to visualize the risks based on the stages of development that refer to the experience of the panelists.

The results of the visualization, then the authors summarize in the form of a percentage provided that if the result is more than 20%, it is considered to meet the requirements as the most dominant risk. While the risk with visualization results below 20% is considered a less dominant risk.

Based on the results of the visualization of the most dominant risks in this study, prevention can later be carried out to overcome the occurrence of risks at each stage of its development. So that the company can plan the system planning carefully and in as much detail as possible in order to avoid project failures due to the risks that arise at each stage of its development [12].

The development stage has six risks, namely R.04 (lack of communication between teams/users and developers), R.06 (lack of expertise/knowledge of the development team), R.07 (unprofessional team), R.08 (Lack of development team) , R.13 (Inadequate facilities), and R.17 (Lack of teamwork). In the implementation phase, there are four risks, namely the risk of R.13 (Inadequate facilities), R.18 (Poor system monitoring), R.19 (Low user acquisition), and R.21 (Untested technology). Meanwhile, in the maintenance phase, there are only three risks, namely R.14 (Lack of user trust), R.18 (Poor system monitoring), and R.19 (Low user acquisition).

Other interesting findings were found from the 17 risks spread over six stages of development, namely there is one risk that is the most dominant and the lowest percentage, the risk is R.01 (lack of analysis of business needs/processes) with a percentage of 37, 14% at the analysis stage. However, R.01 is also the lowest risk percentage at the design stage with only 20.00% gain. And R.01 is also the single most dominating risk in the three stages of development, namely at the planning, analysis and design stages.

IV. CONCLUSION

By using the Delphi method and utilizing the experience of more than 40 panelists involved in software development projects in Indonesia to identify the risks that are considered the most impactful, the differences in

risk for each stage of development and risk mitigation strategies, several interesting findings are obtained which are presented in 3 discussion points, namely :

Of the 22 risks found based on the brainstorming stage, 17 risks reached consensus values and 5 risks did not reach consensus. This is interesting because the 5 risks have a high number of responses and mean values. And other findings show that the 17 who reach consensus have a range of values that are not too far apart. This indicates that these risks have a similar impact on the failure of software development projects. The farthest range of values is in the low user acquisition risk and the risk of cultural differences with a range of values reaching 0.31. Although cultural differences have the lowest average score and are rarely mentioned, cultural differences are one of the risks that have an impact on the failure of software development projects.

Based on the 17 risks that have reached consensus, it shows that R.01 (Lack of analysis of business needs/processes) is the only risk with the highest percentage of the six stages of development, the percentage is 37.14% in the analysis stage and R.01 is also the one. the only dominant risk that exists at the design development stage with a dominant value of 20% or the lowest dominant value. R.01 is also the dominant risk in the three stages of development, namely planning, analysis and design.

Based on the mitigation strategy obtained from the panelists' responses, most of the risks of software development failure can be handled by evaluating and increasing communication, both communication between development team members and communication with users or clients.

Based on research that has been done by previous researchers related to the risk of software development project failure (Table 2.1), this study found several new failure risks such as, Lack of analysis of business needs/processes, Unprofessional team, Lack of user understanding, Inadequate facilities, Lack of user trust, Poor system monitoring, Low user acquisition. The authors also found the risk that the dominant risk difference at each stage of development as well as risk mitigation strategies for the risks that the authors found.

ACKNOWLEDGMENT

We, as researchers, would like to express our deepest gratitude to the Ministry of Education and Culture and Research and Technology through Program Kompetisi Kampus Merdeka (PKKM), and the University of Muhammadiyah Palangkaraya for their support, as well as to LP2M UM. Palangkaraya, and those who have supported researchers to continue working in the field of educational research, hopefully the results of this research can be useful for the community, especially the community. innovators and education experts.

REFERENCES

[1]. The Standish Group International, "CHAOS Report 2015," 2015. [Online]. Available: www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf.

[2]. K. Dewantara, "Identifikasi, Penilaian, dan Mitigasi Risiko Keamanan Informasi Berdasarkan Standar ISO 27001: 2005 dan ISO 27002: 2013 Menggunakan Metode (Studi Kasus: ISNET)," Institut Teknologi Sepuluh Nopember, 2016.

[3]. B. Roy, R. Dasgupta, and N. Chaki, "A Study on Software Risk Management Strategies and Mapping with SDLC BT - Advanced Computing and Systems for Security: Volume 2," R. Chaki, A. Cortesi, K. Saeed, and N. Chaki, Eds. New Delhi: Springer India, 2016, pp. 121–138.

[4]. B.-A. Andrei, A.-C. Casu-Pop, S.-C. Gheorghe, and C.-A. Boiangiu, "A Study on Using Waterfall and Agile Methods in Software Project Management," *J. Inf. Syst. Oper. Manag.*, vol. 13, p. 125+, Oct. 2019, [Online]. Available: https://link.gale.com/apps/doc/A586469729/AONE?u=anon~7d15e9b5&sid=googleScholar&xid=524d4494.

[5]. Y. K. Dwivedi *et al.*, "Research on information systems failures and successes: Status update and future directions," *Inf. Syst. Front.*, vol. 17, no. 1, pp. 143–157, 2015, doi: 10.1007/s10796-014-9500-y.

[6]. N. Dalkey and O. Helmer, "An Experimental Application of the Delphi Method to the Use of Experts," *Manage. Sci.*, vol. 9, no. 3, pp. 458–467, Oct. 1963, [Online]. Available: http://www.jstor.org/stable/2627117.

[7]. L. Giannarou and E. Zervas, "Using Delphi technique to build consensus in practice," *Int. J. Bus. Sci. Appl. Manag.*, vol. 9, pp. 65–82, Aug. 2014.

[8]. W. Widiasih, P. D. Karningsih, and U. Ciptomulyono, "Development of Integrated Model for Managing Risk in Lean Manufacturing Implementation: A Case Study in an Indonesian Manufacturing Company," *Procedia Manuf.*, vol. 4, pp. 282–290, 2015, doi: https://doi.org/10.1016/j.promfg.2015.11.042.

[9]. C. Okoli and S. D. Pawlowski, "The Delphi method as a research tool: an example, design considerations and applications," *Inf. Manag.*, vol. 42, no. 1, pp. 15–29, 2004, doi: https://doi.org/10.1016/j.im.2003.11.002.

[10]. T. M. Maher *et al.*, "Development of a Consensus Statement for the Definition, Diagnosis, and Treatment of Acute Exacerbations of Idiopathic Pulmonary Fibrosis Using the Delphi Technique," *Adv. Ther.*, vol. 32, no. 10, pp. 929–943, 2015, doi: 10.1007/s12325-015-0249-6.

[11]. P. Hopkin, *Fundamentals of Risk Management : Understanding Evaluating and Implementing Effective Risk Management*, 2nd Editio. New Delhi: Replika Press, 2012.

[12]. R. Kaur and J. Sengupta, "Software Process Models and Analysis on Failure of Software Development Projects," *ArXiv*, 2013.