A Reseach study on importance of Testing and Quality Assurance in Softwate Development life cycle (SDLC) Models

[1] Muhammad Junaid, [2] Saleem Zubair Ahmed

[1] Department of Software Engineering the Superior College, Lahore 54700 Pakistan.

[2] Department of Software Engineering the Superior College, Lahore 54700 Pakistan.

[1] Junaidchoudhry12340@gmail.com [2] saleem.zubair@superior.edu.pk

Abstract— nearly all of the software development metrics that are in use today focus on later stages, such as development and testing. However, initial bug detection throughout the SDLC (software development lifecycle) can greatly affect collaboration efficiency, spending less time fixing bugs later and more time preventing them. Additionally, subsequent rework increases the cost of quality and wastes additional time on the development team. The concept of quality is also included in the domain of software development, where it is important to thoroughly validate a software system at various levels of testing. Competition is fierce today and business and platform requirements change frequently, so support and updates must be based on current requirements for long-term and stable use of the software. Software testing is one of the complex activities that any organization undertakes to ensure the value and quality that ensures the viability of software products on the market. This document describes the concept of testing and its role in quality assurance, test cases and test levels, and how tests and tests are planned, implemented, and monitored. The purpose of this overview is to study the current SDLC classification. Initially characterize a set of quality indicators for the software process. In this paper we organized a systematic review that associates with time, cost, challenges and quality of the software product with the SDLC phase.

Keywords—SDLC, Quality metrics, Classification of SDLC, Testing techniques, Importance of SDLC throught cycle

1 INTRODUCTION

Software development practices have been progressing in the past eras. Some Agile practices have been developed in the past few decades that evolve the theory of SDLC and their implementation methods. SDLC is basically defined as the time that could be mandatory for activities that are used throughout the whole process such as defining the project, development of that project, testing of that project, delivery, maintenance and feedback etc. the development crew's efficiency and the quality of the software vary on the usefulness of analyzing and defining the Software throughout then whole process. Initial imperfections discovery can be a key of a successful, interactive and effective project. Though, its phase's classification depends on the practices of company's concerns. Company preferences decides the techniques that are used for evaluation and measuring the quality of Software process. However the set of methodologies that can be traced throughout the evaluation procedure may vary on different concerns. This paper presents the division of SDLC phases, some evaluation practices and different measurements that are existed to test the quality of software. That's why we agreed to organize a systematic literature on SDLC process. This research further guide us in many different ways such as process methods, development stages, methods to detect the efficiency of process etc. In order to further start our research we have led research requirements and questions figured by regarding following queries:

RQ1: In which categories Software development life Cycle phase can be divided?

RQ2: what are the current ways to test quality of software in initial phases?

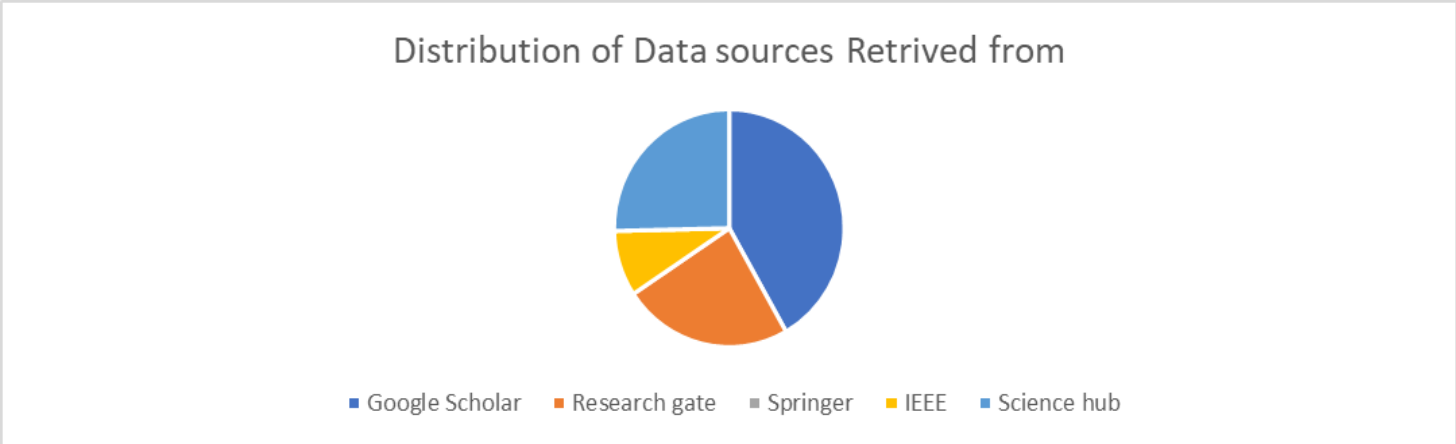
RQ3: what executions are needed in SDLC phases?

RQ4: Throughout the SDLC phase, what measures are required?

2 RELATED WORK

The author Gelperin et al [10] presents the work based on the development of software test engineering. This was tracked by looking at conversions in the testing cycle model and expertise level over the long haul. Binary stage models, for example dual life cycle modes i.e. the evolution and prevention models, , the stock and failure models, have been proposed in order to delineate the development of software testing. Hamlet et al. [11] give more extensive models and more exact outcomes on the connection between partition likelihood, effectiveness and failure rate. In their article, Vishwas Massey and K.J. Satao [12] compares the performance of different SDLC models and suggests new models to improve performance. However, none of the articles compares the search method with the Software development life cycle process. Richardson and Malley [13] presented the first methods of using specifications to select a test case. They proposed specification-based test methodologies, expanding the wide range of implementation-based test methods for application to official specification language. Madjski Leh [14] and his colleagues introduced the concept of using a set of secondary variables and its application to largescale ac-

cessible software with a variety of procedures. The author further demonstrated that secondary mutagenesis techniques can meaningfully enhance the productivity of mutation testing at the cost of testing intensity. Authors who are confident that partition tests are likely to detect flaws at least have the cost of reducing their comparative advantage over randomized tests. In [15] authors analyzed the maturity of their knowledge of the test methods. To this end, they reviewed current empirical research on test methods. To the best of their knowledge, they ranked the test methods and parameters chosen for comparison.



3 RESEARCH METHODOLOGY

We conducted systematic study in order to get the detailed information about SDLC and its quality experience in terms of cost and time. For this purpose, we picked out some research questions on SDLC phases and their matric to carry out the evaluation of related information so as to achieve our focus towards this study.

we surveyed the [1] framework study formation in order to observe the much appropriate and applicable literature that can be expected. onwards we utilize explicit inclusion and exclusion criteria rules for getting to likely essential investigation. The main focus of this study is to get data regarding present types of stages in SDP(Software development Process) in assortment SDLC for primary study. After that we'll talk about the literature terms of value to assess the primary study. we have led research requirements and questions figured by regarding above queries mentioned in Introduction.

These part depicts the arrangement of actions executed to address the formed research question throughout this SLR. In the beginning, the methodology was characterized by regulating the search boundaries to practice for search system. search boundary incorporate "Software Development Life Cycle Phases and quality measurements" and the other is "Software improvement initial life cycle stages and measurements" . As Kitchenham Achimugu [1] Says, we utilized upgrades procedure of determining significant terminologies from the exploration questions. Diverse data bases were chosen in order to select different research papers in terms of enhance the information regarding SDLC phases and quality measurements. The search terms are following.

- Google Scholar
- Research gate
- Springer
- IEEE
- Science hub

We get a yield of in excess of 300 distributions that were accessible in open source libraries that are recorded previously. The consequences of the pursuit terms were arranged into an primary selected concentrates as per the determination guidelines that is proposed in [2]. The methodology incorporates a few stages to direct a subjective appraisal of the distributions. The research is distributed in three phases, in the beginning the underlying arrangement of distributions are selected with programmed and manual search procedures. In the Second step, research papers and related articles were selected as primary search as indicated by their keywords, abstract, title and conclusion.in the last and third step , the primary search were formerly checked in a subtleties by completely auditing the articles. By considering the fact that that the area of initial phases of SDLC isn't completely investigated at this point. Throughout the SLR, we have put In

the exclusion criteria in order to get rid of duplications and those studies which are not completed yet as well as irrelevant articles. . The studies and articles that don't address definite partition of phases of SDLC were excluded.

3.1 Data Collection Assessment

Because of the previously mentioned choice standards, the main choice of studies was led. The primary selection of studies contains a series of keywords addition into information sources. Accordingly, we gathered in excess of 300 publications. The distribution is shown in fig 2. 56% distributions from Google Scholar, 32% publications from Research Gate, whereas 12% distributions from IEEE and 34% publications from Science hub, an open source as some of the articles were not open as free use.

In the initial step of studies determination as indicated by abstract and title, just 32% of studies were acknowledged and 7% were abolished for the reason of duplications. The following stage were led by perusing the entire distribution with subtleties.

3.2 Result Section

This Organized Analysis evaluate Software measurements, prototypes and techniques in the direction of evaluation and investigate the quality of software in initial stages, mainly focused on Design prototypes and Requirements management of SDLC. Along with the primary emphasis continued through embeddings exploration towards open-source information collections, to some extent more than 300 distributions were chosen. Though, next emphasis was based on investigation of Title, Keywords and Abstract. We arranged 75 essential examinations identified with our subject of revenue. In the wake of filtering basically chosen concentrates entire substance in detail, they were arranged from 0-1scale as binary measure in the request for relevance where 0 indicates "not relevant" and 1 presents the information that is relevant. However 9.11% compositions were excluded because of the reason of duplication.

4 Results and Discussions

This Unit presents the results of detailed survey. By conducting the current Software development life Cycle process beginning stages, systems and measurements to survey and assess the nature of the software, we have responded to each of the four exploration questions.

RQ1: In which categories SDLC phases can be divided?

Throughout this Systematic study, we characterized an overall arrangement of stages and a set of measurements that are appropriate to follow the investigation of quality of software; testing of Software [9], of design [17], or overall models [14], likewise thinking about the cycle to gather [6–10], the concealed software Prototype [5-9], the objective framework [21-26], their utilization in forming prototypes [2, 25-29],. By and large, practically the entirety of the examinations expounded software life initial stages into Requirements .The executives and Design stage, and once in a while Code. The distributions weight addressing the distinct primary stages of programming life is portrayed in the accompanying rundown: Requirements stage – 22.2% , Requirements and Design stages – 17.9% , Design – 38.7%, Design and Code stages – 6.7% ,Code and Testing stages – 4.6% , All stages – 20.0% .However, a portion of the papers characterized periods of software metrics in a certain way. As the paper [37] characterizes the SDLC process initial stages incorporates the Classification of the Software solution preliminary Design, User Requirements Analysis and the Peripheral Performance Classification. Though, the initial triple phase's association is known as the Requirements phase. The situation comprises the multitude of activities throughout the deterioration of the product design parts. Although, scholars in [22] set up common SDLC initial stages that are described in the following:

- Initial Planning stage - the specialized and monetary reason for the undertaking ought to be set up

- Analysis - the practical exhibition prerequisites for the Software design issues are characterized. This current stage's outcome is the productive conclusion of PDR which is the abbreviation of Preliminary Design Review

- Prototype – The mentioned stage incorporate the portion of prerequisites to software development segments and finishes by means of CDR known as complete Critical Design Review

In the context of above circumstances, analysis and testing stages are expressed in [35] which followed to requirement Management stage as indicated by the exercises continued during these stages.

RQ2: what are the current ways to test quality of software in initial phases?

Specifically, a few investigations plus different studies recommending various models or ways to deal with evaluation and testing the quality of Softwrae in initial stages. Aversano et al. presented the requirement documentation, in this way the complete evaluation of the credentials of ERP, which is an open source frameworks to comprehend the great citations. The authors divided the quality of Software as far as two perspectives: the first one is Structure Quality and the other one is called Content Quality. The article demonstrates the records are made out of reports that are related to different, APIs. Basically the researcher recommends the Information Extraction for quality assessment. Furthermore, Information Retrieval methods are suggested to create an impartial research [37]. The researchers essentially suggested three ERP frameworks: the first one is Open bravo, the subsequent framework is Compere and the next one is known as Dampier in order to acquire outcomes. The author [39], proposes different quantitative measures in order to match up with the requirement details in addition oth-

er mechanized instruments like Ada, SREM, ISDOS, SADT and so on. Whereas the CAME (Computer Assisted Software Measurement and Evaluation) apparatuses are basically the devices for demonstrating and deciding the measurements of development of software parts mentioning toward the metrics. By and by, the CAME apparatus region additionally incorporates the devices for model-based software development component measurements presentation, introduction of estimation findings, factual investigation and assessment [38]. However the Service Oriented Requirements Traceability Tool (SORTT) creators built up a model device in the direction of measure automation of a single or multiple process. In other words the tool presets or automates the indexing, filtering, querying and some translating and so on. That One motivation is to moderate issues, for example, effort, time for mining evidence linkages [37].

A portion of the research interpret a few bunching investigation methods to calculate the quality of software like k-means and fuzzy c-means, fuzzy mean and so on [12], [40-47]

RQ3: what executions are needed in SDLC phases?

The execution that are needed during the phase of SDLC are some arranged actions throughout the various period of development process. Though, few examinations have incorporated the total presentation of actions that had better to be possible throughout the referenced stages. At this time, there are a few articles containing data about the activities of the mentioned stages. As indicated by [22], the scientific categorization of beginning stages incorporates stages and activities incorporate System requirements, user analysis, System prerequisites, System Design. By and large, the Software development stages include "Requirements Analysis and Definition, and Design stages" [35]. System Analysis plus Definition stage includes exercises similar to prerequisites evocation, requirement investigation, requirement approval, and feasibility study and requirement documentation. Design stage involves various activities, in which the general framework design is set up. This stage includes a few exercises like inspecting the requirements record, picking the architectural plan technique, picking the programming language, verifying, indicating, document design exercises.in the last, we know how to contend that the actions throughout the periods of software measure which rely upon the kind of assessment technique and the quality measurement the organization pick. Notwithstanding, the exercises throughout the overall stages are given as the outcomes from the few concentrates in this segment above.

RQ4: Throughout the SDLC phase, what measures are required?

SDLC software measures are generally corelated with uncertainty in probability that further evaluated FST [49]. The researcher [52] argues that this measure can be evaluated by set theory in order to capture the uncertainty. The similar approach is defined in other studies that characterize the stages and measurements of SP is described in [45] [32][21]. The author [37] directed an exact approval of Object Oriented measurements known as (OO) which is Object oriented analysis and plan technique. The reserchers talked about the connection among Chidamber and Kemerer's OO measurements plus Fault Probability in different stages of the life cycle. Essentially, the author[47] recommends a prototype for the deformity discovery presents in initial phases of SDLC such as plan and initial coding stages know how to be pre-characterized with the complexity and cohesion (CCC) related measures and coupling.

As the author [60] presents the reliability of framework which be capable of chosen by way of planning the prototype. Likewise in [16], the key variables that are involved in requirements of user activity in the executive's stage were characterized as reliability, reaction time, UI, functioning and dependability. Whereas the primary variables of user interest incorporate development time, cost, functions performed, modifiability, reliability, maintainability and dependability. However [42] anticipates that the reliability cannot be predicted because of the reason of computational complexity.

Throughout the study, we have gathered some measurements, regardless, known as McCabe, the other one is known as Halstead, some known as Cyclomatic Complexity and principally the CK measurements were the well known matrices that were referenced in larger part of chosen studies [40-55]. Plus, numerous investigations have been focused on measurements deduction dependent on various perspectives. Mostly Researchers were focused on Module Complexity, Functionality and their maintainability.

As we expressed from the earliest preliminary point, the point of this SLR is to characterize and assess the strategies, their relating measurements and the beginning stages to lead appraisal and assessment of nature of the Software cycle. Research inquiries that is analyzed to characterize the grouping of the SDLC stages, the current prototypes for quality of software in beginning stages, actions which executed throughout the mentioned stages and measurements. First and foremost, we characterized an overall arrangement of stages and accommodating measurements be situated related to the broadly utilized prototypes like Agile techniques. Some of them known as Spiral, waterfall etc. Constraints of exploration is that we grabbed the overall order of SDLC stages instead of grouping them into Spiral and Agile etc. In spite of the way the agile technique stages are acted in a brief phase and sequentially rather than Waterfall prototype. Aftereffects of SLR has demonstrated that 80.0% of the examinations show Design stage and Requirements Management as beginning stages of programming improvement measure. Additional, numerous strategies subsists in terms of survey the quality of Software like: CAME, CCCC, etc. The utilization of Machine Learning approaches [21] investigation of specific modules to examine complexity, usefulness and workability. Some of the exploration contemplates decipher a few bunching investigation methods to anticipate programming measurements quality like k-mean, Gaussian com-

bination model, fuzzy c-mean, and so on. To summarize, chose measurements give more bits of knowledge whether the essential estimation is effectively followed and investigated.

5 Conclusion

Taking everything into account, an efficient project attains the expertise of trace, track and control of the software development process all through the SDLC. To keep up the consistent speed of software development and timing, one necessities to gauge the software interaction as right on time as could really be expected. Generally, the beginning stages incorporate necessities the design phase and requirement management. The measurements can change contingent upon the philosophy and the objective of the organization. The upcoming works toward this path will be situated the further conversation of Probability, Uncertainty level. Some of the experiment on the valuable and working techniques to lead the quality of software assessment measure. According to Benjamin "The bitterness of poor quality remains long after the sweetness of low price is forgotten". Accordingly, one ought to never postpone in assuring the interaction quality that will prompt a resulting item.

References

[1] Ragunath, P.K., et al., Evolving a new model (SDLC Model-2010) for software development life cycle (SDLC). International Journal of Computer Science and Network Security, 2010. 10(1): p. 112-119.

[2] Rani, S.B.A.S.U., A detailed study of Software Development Life Cycle (SDLC) models. International Journal Of Engineering And Computer Science, 2017. 6(7).

[3] Osterweil, L., Strategic directions in quality of software. ACM Computing Surveys (CSUR), 1996. 28(4): p. 738-750.

[4] Jamil, M.A., et al. Software testing techniques: A literature review. IEEE.

[5] Amland, S., Risk-based testing:: Risk analysis fundamentals and metrics for software testing including a financial application case study. Journal of Systems and Software, 2000. 53(3): p. 287-295.

[6] Redmill, F., Theory and practice of risk-based testing. Software Testing, Verification and Reliability, 2005. 15(1): p. 3-20.

[7] Bhatt, D., A Survey of Effective and Efficient Software Testing Technique and Analysis. Iconic Research and Engineering Journals (IREJOURNALS), 2017.

[8] Bertolino, A. Software testing research: Achievements, challenges, dreams. IEEE.H

[9] Gelperin, D. and B. Hetzel, The growth of software testing. Communications of the ACM, 1988. 31(6): p. 687-695.

[10] Hamlet, D. and R. Taylor, Partition testing does not inspire confidence (program testing). IEEE Transactions on Software Engineering, 1990. 16(12): p. 1402-1411.

[11] Kaur, M. and R. Singh, A Review of software testing techniques. International Journal of Electronic and Electrical Engineering, 2014. 7(5): p. 463-474.

[12] Richardson, D., O. O'Malley, and C. Tittle. Approaches to specification-based testing.

[13] Madeyski, L., et al., Overcoming the equivalent mutant problem: A systematic literature review and a comparative experiment of second order mutation. IEEE Transactions on Software Engineering, 2013. 40(1): p. 23-42.

[14] Juristo, N., A.M. Moreno, and S. Vegas, Reviewing 25 years of testing technique experiments. Empirical Software Engineering, 2004. 9(1-2): p. 7-44.

[15] Whittaker, J.A., What is software testing? And why is it so hard? IEEE software, 2000. 17(1): p. 70-79.

[16] Claessen, K. and J. Hughes, QuickCheck: a lightweight tool for random testing of Haskell programs. Acm sigplan notices, 2011. 46(4): p. 53-64.

[17] Harrold, M.J. and G. Rothermel, Performing data flow testing on classes. ACM SIGSOFT Software Engineering Notes, 1994. 19(5): p. 154-163.

[18] Mouratidis, H., P. Giorgini, and G. Manson, When security meets software engineering: a case of modelling secure information systems. Information Systems, 2005. 30(8): p. 609-629.

[19] Ahmad, Z., et al., Implementation of Secure Software Design and their impact on Application. International Journal of Computer Applications, 2015. 120(10).

[20] Shoemaker, D. and N.R. Mead, Evaluating Software Assurance Knowledge and Competency of Acquisition Professionals. 2014, CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST.

[21] Offutt, J., Quality attributes of web software applications. IEEE software, 2002. 19(2): p. 25-32.

[22] Pohl, C. and H.-J. Hof, Secure scrum: Development of secure software with scrum. arXiv preprint arXiv:1507.02992, 2015.

[23] Shreyas, D. Software engineering for security: Towards architecting secure software.

[24] Saba, T., et al., Annotated comparisons of proposed preprocessing techniques for script recognition.

[25] eural Computing and Applications, 2014. 25(6): p. 1337-1347.

[26] Essafi, M. and H.B. Ghezala, Meta-modeling based secure software development processes. International Journal of Secure Software Engineering (IJSSE), 2014. 5(3): p. 56-74.

[27] Devanbu, P.T. and S. Stubblebine. Software engineering for security: a roadmap.\

[28] Shin, M.E. and H. Gomaa, Software requirements and architecture modeling for evolving non-secure applications into secure applications. Science of Computer Programming, 2007. 66(1): p. 60-70.

[29] Arbain, A.F., I. Ghani, and S.R. Jeong, A systematic literature review on secure software development using feature driven development (FDD) agile model. Journal of Internet Computing and services, 2014. 15(1): p. 13-27.

[30] McGraw, G., Software security. IEEE Security & Privacy, 2004. 2(2): p. 80-83.

[31] Wongthongtham, P., et al., Development of a software engineering ontology for multisite software development. IEEE Transactions on Knowledge and Data Engineering, 2008. 21(8): p. 1205-1217.

[32] Bukhari, Z., J. Yahaya, and A. Deraman, A Conceptual Framework for Metrics Selection: SMeS. International Journal on Advanced Science, Engineering and Information Technology, 2018. 8(6): p. 2294-2300.

[33] Jones, R.L. and A. Rastogi, Secure coding: building security into the software development life cycle. Inf. Secur. J. A Glob. Perspect., 2004. 13(5): p. 29-39.

[34] Daud, M.I. Secure software development model: A guide for secure software life cycle.

[35] Bokhari, M.U. and S.T. Siddiqui, TSSR: a proposed tool for secure software requirement management. International Journal of Information Technology and Computer Science (IJITCS), 2014. 7(1): p. 1.

[36] Nodehi, A., et al., Intelligent fuzzy approach for fast fractal image compression. EURASIP Journal on Advances in Signal Processing, 2014. 2014(1): p. 112.

[37] Islam, S. and P. Falcarin. Measuring security requirements for software security. IEEE.

[38] Islam, S., H. Mouratidis, and J. Jürjens, A framework to support alignment of secure software engineering with legal regulations. Software & Systems Modeling, 2011. 10(3): p. 369-394.

[39] McGraw, G., Building secure software: A difficult but critical step in protecting your business. Cigital, White Paper, available at: http://www. cigital. com/whitepapers, 2003.

[40] Khan, A.A., S. Basri, and P.D.D. Dominic. A propose framework for requirement change management in global software development. IEEE.

[41] Niazi, M., et al., GlobReq: A framework for improving requirements engineering in global software development projects: Preliminary results. 2012.

[42] Jiménez, M., M. Piattini, and A. Vizcaíno, Challenges and improvements in distributed software development: A systematic review. Advances in Software Engineering, 2009. 2009.

[43] Lopez, A., J. Nicolas, and A. Toval. Risks and safeguards for the requirements engineering process in global software development. IEEE.

[44] Gomes, V. and S. Marczak. Problems? We all know we have them. Do we have solutions too? A literature review on problems and their solutions in global software development. IEEE.

[45] Harikesh Bahadur Yadav and Dilip Kumar Yadav. Construction of membership function for software metrics. Procedia Computer Science, 46:933–940, 2015.

[46] Bingbing Yang, Qian Yin, Shengyong Xu, and Ping Guo. Quality of software prediction using affinity propagation algorithm. In 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence). IEEE, June 2008.

[47] Fred van den Bosch, John R. Ellis, Peter Freeman, Len Johnson, Carma L. McClure, Dick Robinson, Walt Scacchi, Ben Scheff, Arndt von Staa, and Leonard L. Tripp. Evaluation of software development life cycle. ACM SIGSOFT Software Engineering Notes, 7(1):45–60, January 1982.

[48] Chandan Kumar and Dilip Kumar Yadav. A method for developing node probability table using qualitative value of software metrics. In Proceedings of the 2015 Third International Conference on Computer, Communication, Control and Information Technology (C3IT). IEEE, February 2015.

[49] Pongtip Aroonvatanaporn, Thanida Hongsongkiat, and B. Boehm. Improving software development tracking and estimation inside the cone of uncertainty. 2012.

[50] Wouter Tengeler. Cone of uncertainty for agile projects, 2014. Online: http://www.themotionstudio.nl/en/cone-of-uncertainty-for-agile-projects/, on 5th feb 2021.

[51] Stefan Luyten. The cone of uncertainty and how to avoid it turning into a wormhole, 2014. Online: https://medium.com/@stefanluyten/the-cone-of-uncertainty-82d21e99fcc2, on 5th feb 2021.

[52] V.R. Basili, L.C. Briand, and W.L. Melo. A validation of object-oriented design metrics as quality indicators. IEEE Transactions on Software Engineering, 22(10):751–761, 1996.

[53] Prathipati Ratna Kumar and G.P Saradhi Varma. A novel probabilistic-ABC based boosting model for software defect detection. In 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS). IEEE, March 2017.

[54] R. Bharathi and R. Selvarani. A framework for the estimation of oo software reliability using design complexity metrics. In 2015 International Conference on Trends in Automation, Communications and Computing Technology (I-TACT-15), pages 1–7, 2015.

[55] Dewanne M. Phillips, Thomas A. Mazzuchi, and Shahram Sarkani. An architecture, system engineering, and acquisition approach for space system software resiliency. Information and Software Technology, 94:150–164, February 2018.

[56] R Bharathi and R. Selvarani. A framework for the estimation of OO software reliability using design complexity metrics. In 2015 International Conference on Trends in Automation, Communications and Computing Technology (I-TACT-15). IEEE, December 2015.

[57] Narimane Zighed, Nora Bounour, and Abdelhak-Djamel Seriai. Comparative analysis of object-oriented software maintainability prediction models. Foundations of Computing and Decision Sciences, 43(4):359– 374, December 2018.

[58] Yue Jiang, Bojan Cuki, Tim Menzies, and Nick Bartlow. Comparing design and code metrics for quality of software prediction. In Proceedings of the 4th international workshop on Predictor models in software engineering - PROMISE '08. ACM Press, 2008.

[59] Sun-Jen Huang and Richard Lai. Deriving complexity information from a formal communication protocol specification. Software: Practice and Experience, 28(14):1465–1491, December 1998.

[60] Jindal, T. (2016). Importance of Testing in SDLC. International Journal of Engineering and Applied Computer Science (IJEACS), 1(02), 54-56.