Factors Contributing in Failures of Software Projects

Article *in* Optical Memory and Neural Networks · May 2019

CITATIONS
3

READS
910

7 authors, including:

Dr Muhammad Hamid
University of Veterinary and Animal Sciences
27 PUBLICATIONS **104** CITATIONS
SEE PROFILE

Furkh Zeshan
COMSATS University Islamabad
22 PUBLICATIONS **217** CITATIONS
SEE PROFILE

Adnan Ahmad
Khyber Medical College
22 PUBLICATIONS **127** CITATIONS
SEE PROFILE

Esma Aimeur
Université de Montréal
186 PUBLICATIONS **1,809** CITATIONS
SEE PROFILE

Factors Contributing in Failures of Software Projects

Muhammad Hamid[†,††], Furkh Zeshan[†††], Adnan Ahmad[†††] and Esma Aimeur[††††]

[†]NCBAE, Lahore, Pakistan
[††]UVAS, Lahore, Pakistan
[†††]COMSATS University Islamabad, Lahore Pakistan
[††††]Department of Computer Science, University of Montreal, Quebec Canada

Summary
Software Project management (SPM) is a vital concern for software industries to follow best practices for successful project completion. Despite the rich availability of SPM literature, every year around 70% of projects cannot gain successful completion worldwide. Software failure impacts the software industry in terms of reduced revenue, development teams with stress and reduced motivation, general population in terms of jobs reduction and the whole country in terms of reduced exports. This study explores the literature on SPM with the objectives of identifying major contributing factors in software failure. The current study, identified 2171 research studies out of which 68 have been thoroughly analyzed, after applying guidelines of inclusion and exclusion. The analysis of 68 selected research papers highlighted 13 influencing factors toward software project failure, with four major, five significant and four insignificant factors, where the major factors are incorrect cost and time estimation. The analysis included 35.29% empirical studies, 47.06% general literature review and 17.65% case studies. The analysis also reflected that 86.77% papers only examined the state of the art while only 13.23% of research studies discussed some algorithm to reduce failure. Further, the analysis found that only 4.41%, studies developed some automation tool for reducing some failure factor while 95.59% of studies did not developed any tool. The findings of this study provide future insights for SPM research as well as the software industry to increase the ratio of successful projects.

Key words:
Software project management, Software development, Software failure factors

1. Introduction

The examination and categorization of project failure have been the subject of extensive study in recent years [1-6]. A software project is considered as failure, when it does not deliver within allocated time, budget or minimum quality [7]. There are many reasons behind software project failure such as lack of project planning [8, 9], scope creeping [10-12], wrong estimation [8, 13-15], incomplete requirements [16], inadequate selection of human resource [17-19], and lack of user involvement [20, 21], etc. As software projects are the primary source of revenue generation for most software companies [22], their failure negatively affect the company's image, goodwill, revenue drive and perceived satisfaction of customers and clients [23]. On the other hand, successful completion of a software project positively impacts on software exports, which, in return, not only enhances the economy of a country but also creates new jobs [24]. For many years, it remains a challenging task for researchers and practitioners to manage information technology (IT) projects successfully. Two major software project management research groups, GAO (established 1979) and Standish Group (established 1994), publish their annual reports regarding software project failure and success [13, 25]. The Standish Group gathers statistics from an extensive databases of projects executed every year and releases report [13], which demonstrates various reasons for software project failure and success. They classify all projects in three major types with respect to their completion status [26], detail is under:

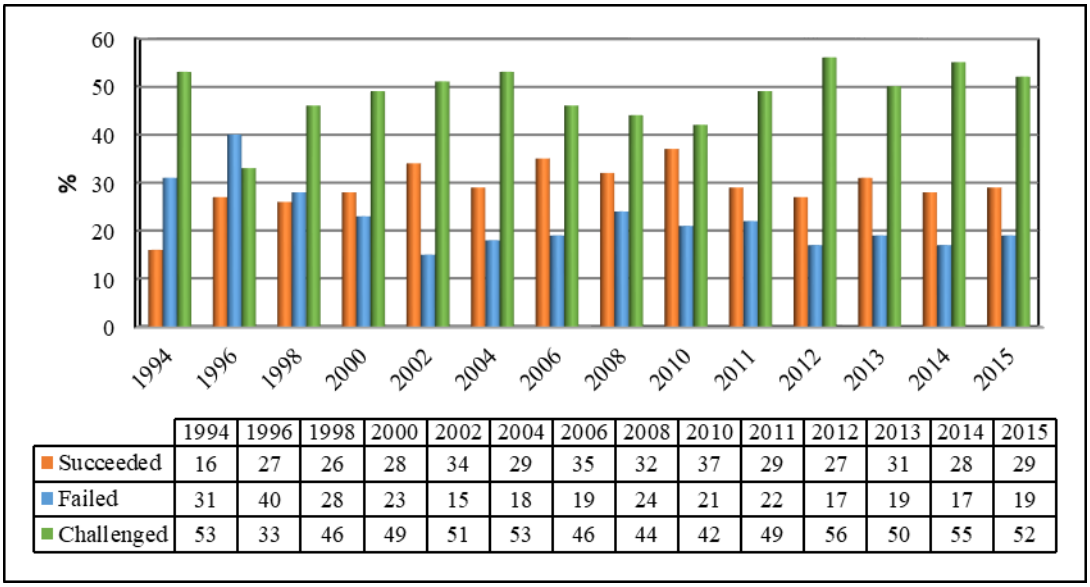
(i) Type 1 (successful) projects are accomplished within allocated time and budget frame along with all already specified functions and features.
(ii) Type 2 (challenged) projects are completed and functional, but exceeded time limits, over-budget, or supported fewer features than specified.
(iii) Type 3 (failed) projects are disproved at a particular point before their completion.

According to Standish group recent findings [13], only 29% projects are successfully completed, 19% projects failed outright; while 52% projects overrun of time, compromised functionality or cost. In this research, both type 2 and type 3 projects were considered as failures. A summary of the Standish group reports throughout the previous 22 years (from 1994 to 2015) is exhibited in Figure 1.

This research aims to investigate various factors that contribute to software projects failure. For this purpose, the literature is identified, assessed and analyzed to explore essential guidelines for avoiding software projects failure in future. This exploration comprised of three phases: a) planning the investigation process, b) conducting the software project failure review, and c) reporting the review results. The outcome shall exhibit the complete perspective of software project failure through identifying failure factors, their ranking, current state of

Manuscript received May 5, 2019
Manuscript revised May 20, 2019

the domain as well as proposing guidelines to overcome these failures.



	1994	1996	1998	2000	2002	2004	2006	2008	2010	2011	2012	2013	2014	2015
Succeeded	16	27	26	28	34	29	35	32	37	29	27	31	28	29
Failed	31	40	28	23	15	18	19	24	21	22	17	19	17	19
Challenged	53	33	46	49	51	53	46	44	42	49	56	50	55	52

Fig. 1 Standish figures from 1994–2015 publication years

This paper is systematized as follows: Section 2, describes state of the art in the context of software project failure. Section 3, presents the systematic literature review process for identification of software failure. Section 4, provides the detailed analysis while Section 5, explains the recommendation and research contribution. Section 6, concludes the paper with main findings and potential future research directions.

2. State of the Art

Various studies have explored the literature in order to extract responsible factors for software project failure. This section provides an overview of the existing research studies to identify existing research in this specific research area, questions raised and interpreted phenomenon of interest.

Jorgensen and Shepperd [27] identified 76 major journals of the field and shortlisted 304 research papers from them for the identification of key software project failure factors. Their analysis concluded that cost estimation is the main factor influencing the project failure. Dikert and colleagues [28] conducted an exploratory review of the literature for identifying factors that affect the software failure. Total 1875 papers were found on keyword search, while 52 research papers were selected for thorough analysis, after applying inclusion and exclusion criteria. Their analysis showed that the major influencing factors are the wrong estimation of time, lack of management support and inadequate human resource. Also, in 2016, Idri and colleagues [29] conducted a review of the literature for the identification of influencing factor towards software failure during 2000 to 2016 years. An analysis of 24 research papers highlighted that estimation of time and cost were the main contributing factors during these 16 years. Guillaume [30] explored causes of software project failure through a case study of different software development circumstances. His case study of 202 software projects identified various software project failure factors including unrealistic project goals, lack of resources, executive support and appropriate planning. Walia and Carver [16] conducted a review in software project development literature to explore the major reason behind software failure. After reviewing 149 papers, they found that inappropriate requirement management is the major influencing factor in project failure. Similarly, Hossain and colleagues [31] explored the literature for the same reason. They found 336 research articles in the initial search, and a final count of 20 research articles were analyzed. The analysis showed that in distributed projects, coordination and communication among team members is a key challenge which contributes towards most project failures. Inayat and colleagues [32] conducted an exploratory review on identifying the failure rates of software projects. The review conducted on literature published between 2002 and June 2013, where an analysis of the finalized 21 papers suggested that improper requirement gathering was a main contributing factor towards project failure during those 12 years. Gupta and colleagues [33] conducted a case study on globally distributed projects. The evidence indicated that the main challenges in such projects execution were cooperation

between team members and estimation. These challenges further lead to project delays or budget overrun. Cerpa and colleagues [34], after an extensive literature review, identified that three major factors are responsible for project failure. These factors include the capability of project manager, unrealistic project plan and inadequate human resource. Komchaliaw and Wongthongtham [35] carried out an extensive review of the literature and identified key factors that lead to software project failure. These factors included inadequate staff, improper project planning and inappropriate requirements. In a case study, Attarzadeh and Ow [36] developed a questionnaire to investigate the factors that lead to software project failure. The data collected from 50 developers suggested that poor planning and scheduling are the main reasons for software project failure. Damasiotis and colleagues [37] conducted an extensive literature review for the identification of crucial failure factors in software development. According to their finding, the wrong estimation of the time is one of the leading reasons behind software project failure.

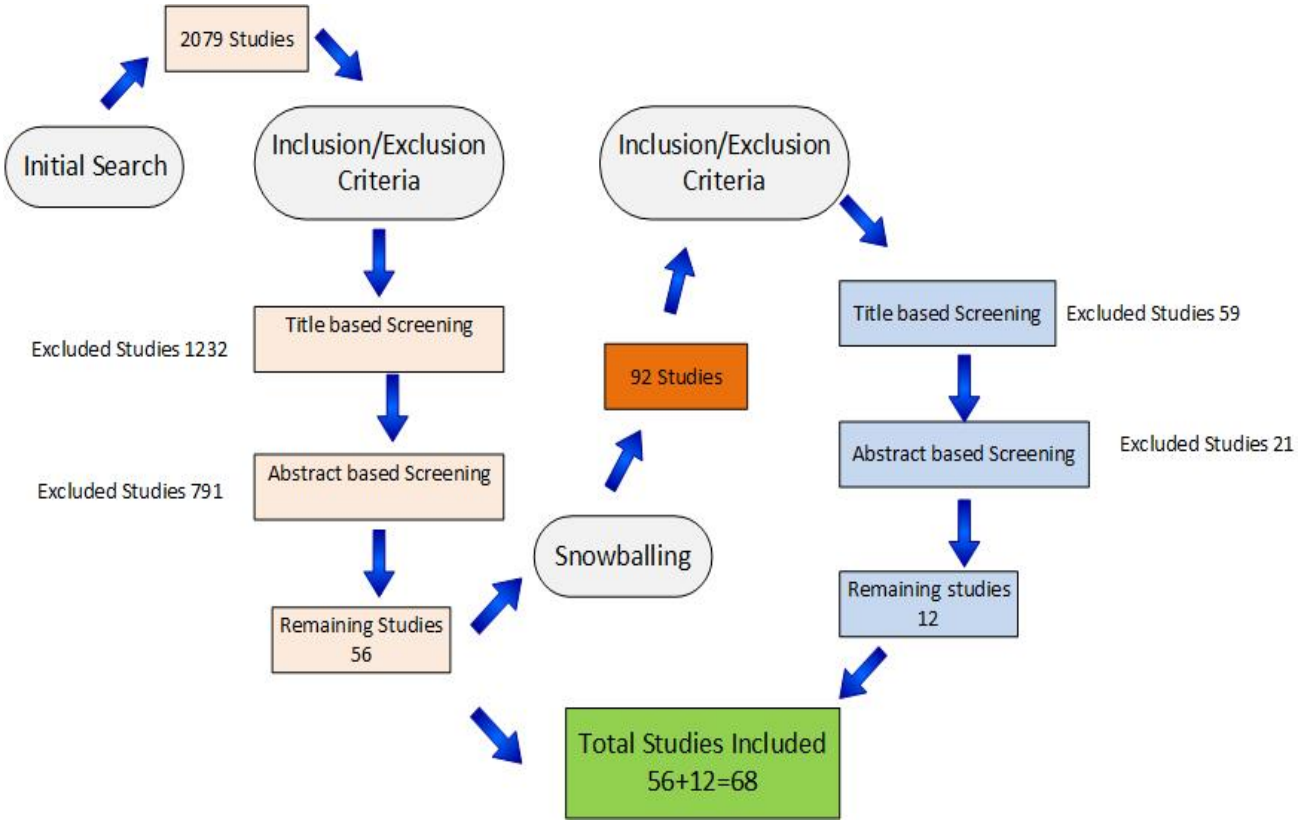


Fig. 2 Overview of the search process.

The above overview have explored various project influencing failure factors, identified in previous research work. However, the existing studies do not provide any insight on the categorization of important failure factors according to 80/20 rule [111], nor they explored the impact of existing automation tools and algorithms to highlight their effectiveness and shortcomings. This study has explored the latest work conducted in this area, along with some neglected issues, which could help researchers and software project management community to reduce their risks of software project failure.

3. Investigation of software failure factors

The literature review provides essential guidelines to identify, assess and analyze available research regarding a specific research questions [38, 39]. This process results in a secondary study, whereas, individual studies that contribute towards it are considered primary studies. This exploratory study adapted the protocol for executing the review in software engineering suggested by Kitchenham and colleagues [38], which consists of three major phases, i.e., planning, conducting and reporting. Explicit explanation of these phases for executing the investigation

process for identifying software failure factors is described below.

3.1 Planning the Investigation Process

This is the first phase of the investigation which consists of a) justification for performing the investigation; b) research question design; c) defining the search strategy; and d) developing inclusion and exclusion criteria.

3.2 Justification for the investigation

There are various needs for performing this investigation of the literature. Its primary aim is to collect the literature on project failure in a systematic way and summarize all prevailing practices and information in an organized and non-prejudiced way. Further, it aims to assess the impact of project failure research and to highlight various challenges that can give more insights for future investigations. This research work will help to provide the latest research findings to the software industry as well as researchers interested in the software development process over some overlooked issues.

3.2.1 Research Questions Design

Developing appropriate research questions is an essential task before conducting the investigation because research questions provide probing directions to extract information from primary studies [38]. To define research questions for this study, the Goal-Question-Metric (GQM) approach [40] was used. This approach starts with identifying specific goals like object, viewpoint, issue, etc. Then, these goals are further refined into various questions and then subdivided into metric [40]; which give ways to get the answers to defined questions. Finally, based on these answers, data can be investigated to evaluate the achievement of specified goals. To identify the failure factors of software projects, following questions were designed by adapting the above-mentioned procedure.

RQ1. What are the main factors for the failure of software projects?
RQ2. Explore various methodologies used in literature to probe the issue?
RQ3. How many algorithms or automation tools were proposed to assist the project managers?
RQ4. What are the limitations of the existing research studies?

3.2.2 Search Strategy

The purpose of defining the search strategy is to find an extensive and unbiased method for the gathering of research material related to research questions. The search strategy was created to maximize the probability to search the related studies in a research area. For search purpose, popular databases used in software engineering research, as described in [41], were used, including IEEE Digital Library, ACM Digitial Library, SpringerLink, and ScienceDirect. The selected databases are extensive and contain bibliographic data of production from all key publisher of the computing literature. In this manner, we utilized the above four databases along with Google Scholar for verifying the collected results and performing some meta-investigations. Our search keys stem from the research questions, while the boolean operator of "OR" and "AND" were utilized to join other words. The search string used in this review was (software OR application OR product OR project) AND (reason OR cause OR factor) AND (failure).

3.2.3 Inclusion and Exclusion Criteria

Inclusion and exclusion criteria are a set of predefined characteristics used to identify research articles to be included in a research study. Inclusion criteria, along with exclusion criteria, make up the selection or eligibility criteria used to rule in or out the target research articles for a research study. The inclusion and exclusion criteria to address the research questions of the current investigation are shown in Table 1.

Table 1: Inclusion and exclusion criteria used in this investigation

Type	Description
Inclusion	Only those papers were considered which, 1. Belong to popular computing research databases, including ACM, IEEE, Elsevier, Springer and Google Scholar. 2. Discussed some failure factor in software project development. 3. Investigated the factors influencing software project failure. 4. Provided evidence of software project failure through case studies, experience reports and field studies. 5. Are published during the years 1990 to 2017.
Exclusion	Exclude the papers, which 1. Belong to non-indexed journals, books, master or doctoral dissertations and the papers that did not undergo a proper investigation process. 2. Belong to hardware or others fields rather than software engineering. 3. Are not relevant to the research questions. 4. Concern news issues or related experience in software project failure. 5. Is not written in the English language. 6. Are duplicate (select the latest version).

3.3 Conducting the Investigation

To conduct the investigation, procedures and instructions defined in the first phase were executed. This phase comprises of: a) selection of primary studies, and b) the quality assessment of the selected studies. The selection of primary studies at different steps of quality assessment is shown in figure 2.

3.3.1 Primary Studies Selection for Investigation

This research was carried out between June 2018 to December 2018. In the initial phase, 2079 studies were extracted using a search string. Through title base screening, 1232 studies were eliminated, reducing the remaining number to 847. Then the abstract based screening was performed on remaining 847 papers and 791 more papers were excluded, reducing the final count to 56 papers. Two researchers, working autonomously, extracted the relevant research papers by scanning portions of every article, like the title and abstract. It was observed that repetition of data was present due to various reasons. For example, some authors published a new version of their previously published work, or a shorter version is published in a conference or workshop. So, all duplicated results were excluded carefully, and only the recent work without duplication was kept. On these 56 papers, snowballing was applied and 92 more papers were identified. The inclusion and exclusion criteria were applied on these newly identified papers, where duplicated, title based screening (92-59=33) and abstract based screening (33-21=12) left 12 papers to be included, making a total of 68 (56+12) papers for thorough investigation. These 68 papers from January 1999 to December 2018 were screened and retained for in-depth analysis. Statistics of selected research studies related to software project failure is given in Table II. Overview of the division of chosen studies as per year is provided in Figure 3.

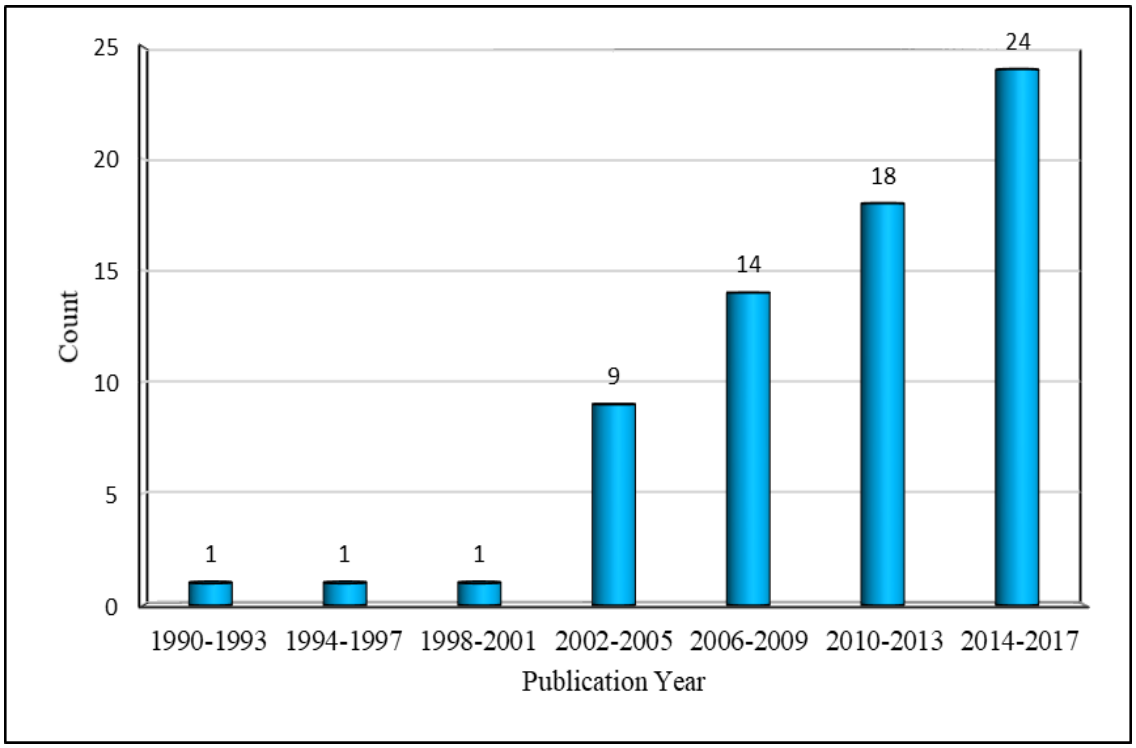


Fig. 3 Selected Papers by Publication Year

3.3.2 Quality Assessment

The quality assessment criteria were used to determine the rigorousness and credibility of the used research methods and the relevance of the studies. Quality assessment was developed to gain the significant results of studies. Only reputed scientific databases were selected in the study.

Table 2: Selected research works Statistics for software project failure

Serial No.	Publication type	No. of Research studies
1	Journal	39
2	Conference	25
3	Chapter	4
	Total	68

4. Analysis of Literature

In this section, outcomes of the in-depth investigation about software project failure factors are summarized to determine the existing gaps and current research trends.

4.1. RQ1. What are the main factors for the failure of software projects?

The objective of this question was to identify the main reasons behind software project failure. Similar explorations have been done earlier in this domain, but to complete the analysis for other overlooked issues, it was important to consider some most recent findings. Although some of the factors are already explored, however, this study organized them in various categories according to their influence on project failure. The analysis of 68 selected research papers highlighted 13 influencing factors that contribute toward software project failure. Among them 4 are major (top 33%), 5 significant (medium 33%) and 4 are insignificant (bottom 33%) factors, as illustrated in figure 4. Out of 68 research papers, 43 highlighted that wrong estimation of time and cost are major factors for software project failure. Time and cost estimation need detailed information about human resources availability, scope and requirements of the software project. Software estimation plays a vital role in software project development as a slight miscalculation not only delays the completion of a software project but also increases its cost. It is observed in various studies that useful software project planning and management are challenging to achieve without proper estimation [8, 9]. The estimation is used by the project manager for managing and controlling a software project. The wrong estimate not only delays the completion of a software project but also increases its cost [8, 14, 42].

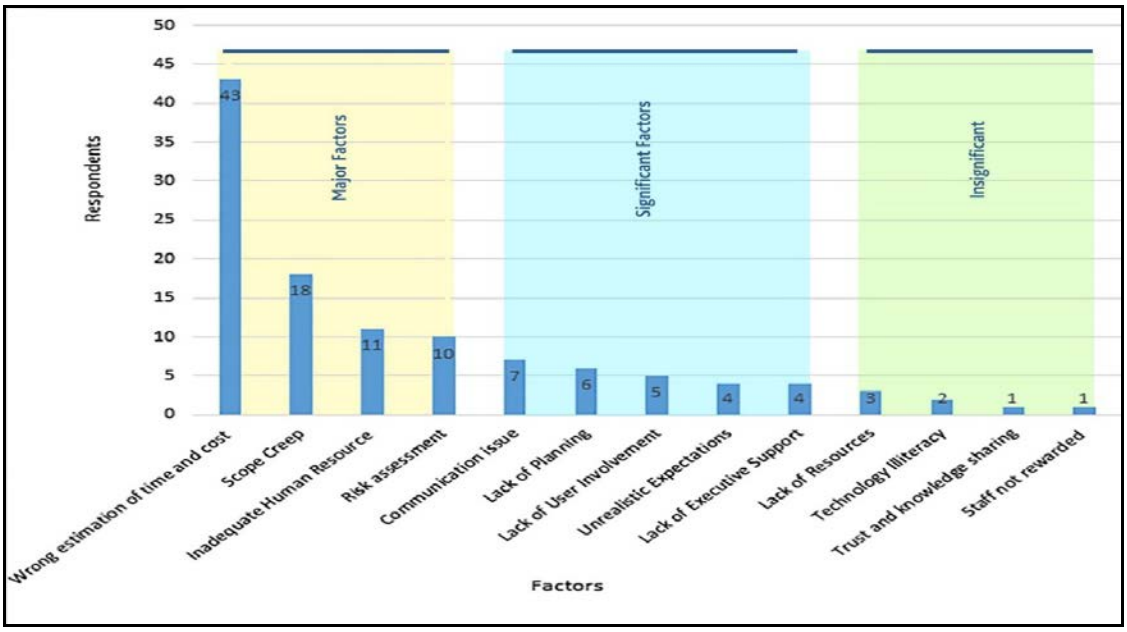


Fig. 4 Project key failure factors

After time and cost estimation, the second major contributing factor is scope creep, highlighted by 18 papers. Scope creeping appears due to changes in requirements at later stages. The change in requirements can occur due to several reasons including the lack of client involvement, or unclear project vision. To handle this issue, software development team has to rework which negatively affects the development cost and time. Ebert and De Man [43] found that one of the causes of scope creep is requirements uncertainty. They identified this after performing an empirical study over 246 software projects from years 2002 and 2003.

Inadequate human resource is the third main reasons behind software project failure. Human resource refers to less trained project managers or lack of project management skills to implement theoretical principles in practice [44]. It also relates to poor selection of IT skills professionals, as well as lack of proper communication skills which often lead to project failure [44]. For example, if the team members are not suitable, a lot of rework may be needed to finalize the project, which ultimately delays and over-budget a software project [45]. On the other hand,

suitable selection of the team members positively impact the software quality [45] and implies the satisfaction of team members which is directly proportional to software productivity [42].

Poor risk assessment is found to be the fourth major reason for software project failure. Proper risk assessment plays an important role in reducing the probability of software failure. Therefore, to achieve a successful outcome, the project manager must identify, assess, prioritize, and manage all the major risks [46]. Verner and colleagues [47] analyzed 70 failed software projects, and identified 57 essential factors that can affect software project failure, where the major reason was poor risk assessment.

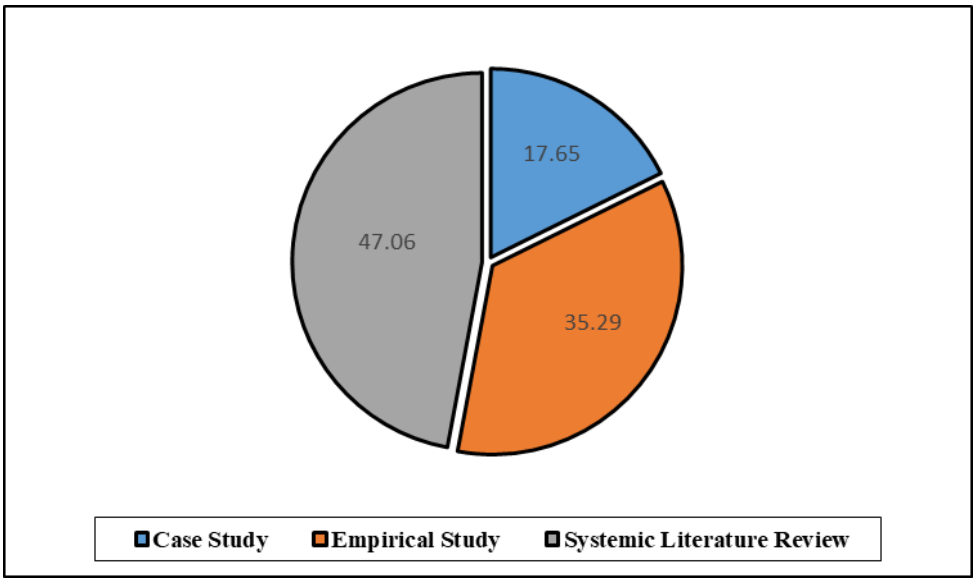


Fig. 5 Types of studies

After these four major failure factors for software projects, various other significant factors were also identified. 7 out of 68 research papers suggested that communication gap between team members during software development is a reason of software project failure; 6 research articles argued for lack of appropriate planning, 5 research articles suggested lack of user involvement, 4 claimed that lack of executive support while 4 found that unrealistic clients' expectations are the significant cause behind project failure.

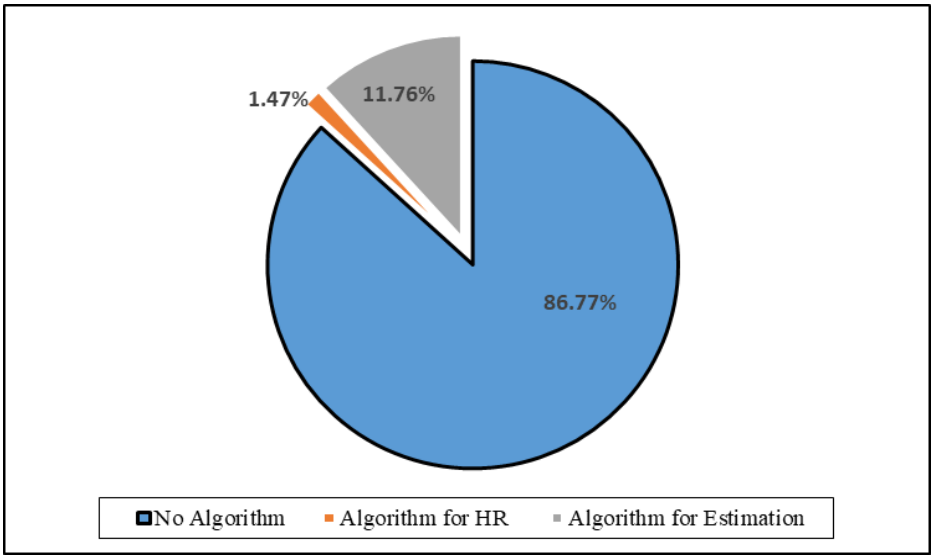


Fig. 6 Level of the estimation algorithm

After the major and significant factors, some other factors were also highlighted by some studies in the literature. We consider them insignificant as they were highlighted by less than 5% of the literature. These factors include lack of resource [16] , technology illiteracy [46], unrewarded human resources [47] and trust among the team members [35]. These low contributing factors might have some impact on the software project failure, but strong empirical evidence will be required before considering them significant. The analysis is summarized in figure 4 along with various categories of failure factors for software projects.

4.2 RQ2. Explore various methodologies used in literature to probe the issue?

This study analyzed 68 articles from literature to synthesize and identify 13 (4 major, 5 significant and 4 insignificant) software project failure factors. Different studies have used different methodologies to identify similar factors. Figure 5, presents an overview of the methodologies followed in literature, where SLR is the major adapted methodology (used by 47.06% articles), followed by empirical study (35.29%) and case study (17.65%).

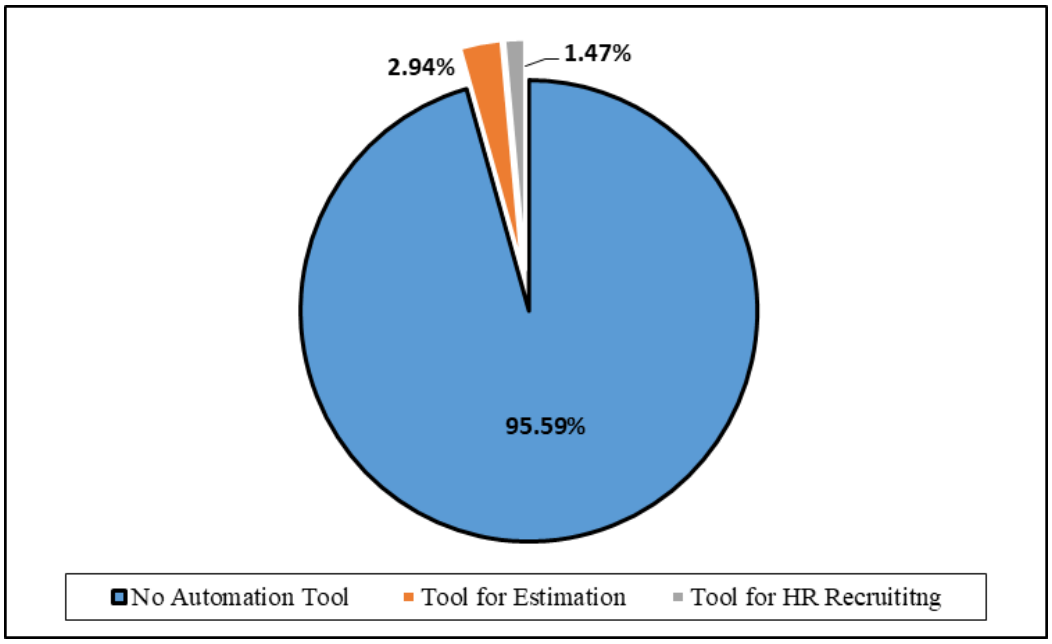


Fig. 7 Level of Automation achieved

Methodology proposed by [38] is used for a complete overview of all the available evidence about a particular domain [50]. Based on the evidence, it can provide a definitive answer to a specific question. SLR reviews the literature in a certain field, giving an outline of the field, recent progress, main problems and challenges the field is still faceing [51]. On the other hand, the empirical methodology is used for acquiring knowledge by means of direct and indirect observation or experience and perception [52]. An empirical methodology investigates a contemporary phenomenon within its real-life context; when the boundaries between phenomenon and context are not clearly evident; and in which multiple sources of evidence are used [53]. The case study methodology is primarily used for exploratory purposes [54, 55].

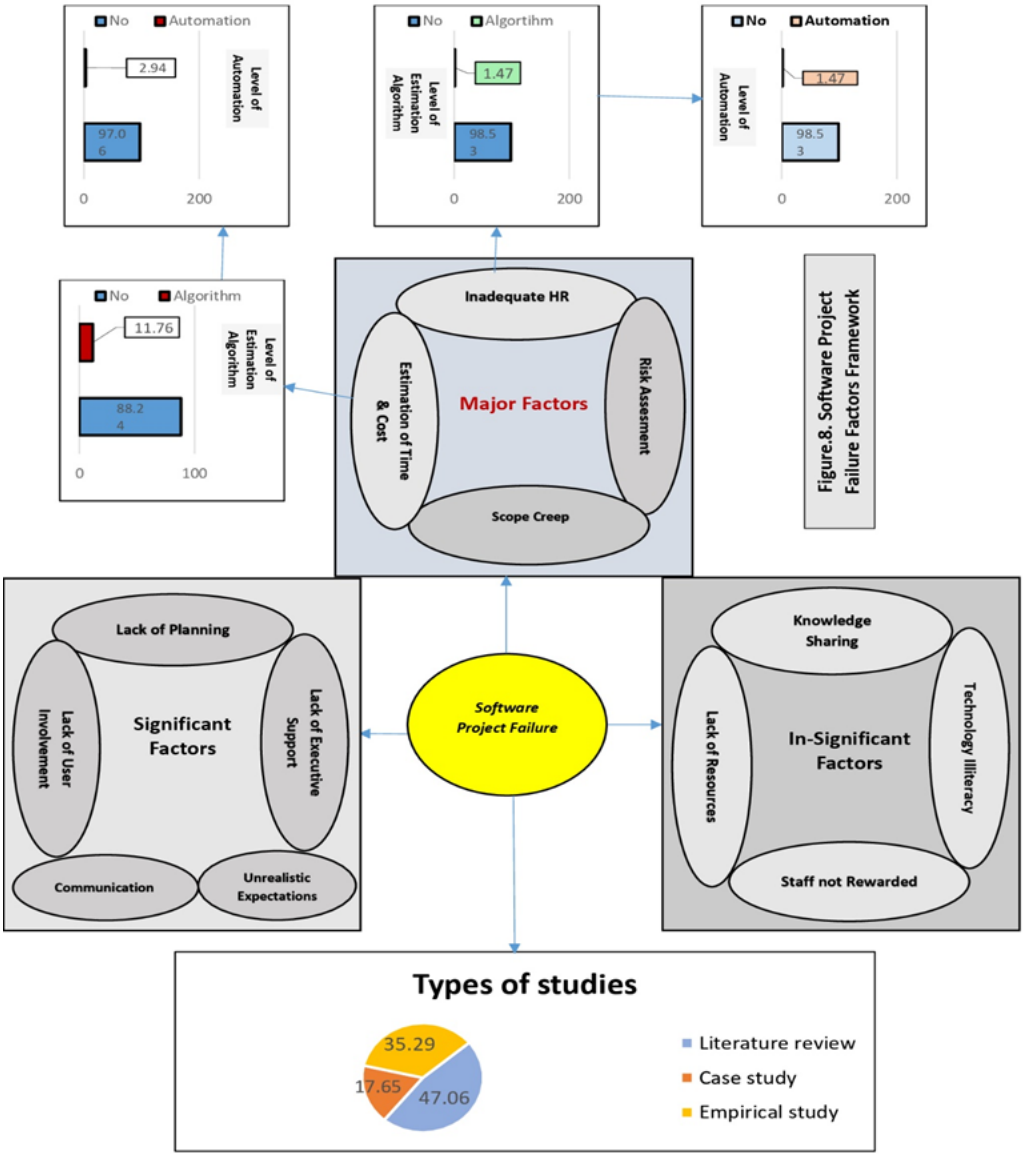
In our search results, around half of the studies adapted the SLR methodology to identify software project failure factors. The main reason behind this might be a broader insight provided by SLR which lacks in empirical study or case study. Both the latter options only explore a part of the domain while SLR offers a more thorough insight into multiple projects with various perspectives of different regions under different circumstances.

4.3 RQ3. How many algorithms or automation tools were proposed to assist the project managers?

The literature analysis showed that only 13.23% papers proposed some algorithm, as shown in figure 6. Likewise, the estimation achieved through automation was found in only 4.41% studies, while the remaining 95.59% research articles did not mention any automation tool as reflected in figure 7. Total 9 algorithms were proposed in the literature for software project failure factors, out of which only 3 were developed as some tool. A summary of these six algorithms and three automation tools are briefly discussed in the sections below, so they may provide some directions for future researchers and practitioners.

4.3.1 Proposed Algorithms

Better decision-making in scheduling and accurate management of the development team according to their skills and experiences are two key areas for a software organizations. Stylianou and Andreou [56] proposed an algorithm for better scheduling and management of team activities to achieve the desired results during software project development. In scheduling constraints, the dependencies among the tasks are highlighted and the required tasks are assigned to team members according to their experience and skills. Similarly, Bahlerao and Maya [57] proposed an algorithm (Generalized Estimation Method (GEM)) to handle the influencing factors of software projects in agile context. Their proposed



algorithm estimates the cost and time of future software projects. Likewise, Thamarai and Murugavall [24] proposed an algorithm (DEAPS) for software cost and time estimation. Their proposed approach used an analogy to retrieve relevant software projects from a database and predicted the cost and time for new software projects.

Along the same lines, Bia and colleagues [58] proposed an approach (Digitization Costs Model (DiCoMo)) to estimate the cost and time for new projects. They discovered various missing factors in existing approaches like COCOMO [59], function points [60], and historical data [61], and proposed an algorithm to minimize these factors through digitization. Additionally, Francisco and

colleagues [62] developed an algorithm to identify risk factors which are more influential in determining project outcome. Their algorithm was tested with several existing software risk prediction models and showed an improved estimation of time and cost. Moreover, Peretz and Opher [63] proposed an algorithm to estimate the time and cost of software projects at early stages of their development lifecycle. For this purpose, they combined the Mk-II Function Points method [64] for software estimation with the ADISSA (architectural design of information systems) [65]. Their results showed better prediction of effort and time.

4.3.2 Developed Automation Tools

Adnan and Afzal [66] developed a tool to estimate the time of agile-based projects and to use the knowledge acquired during software development for upcoming software projects. The proposed tool uses the knowledge base and multi-agent system for estimation and saving the lesson learnt during software development. In another approach [46], Jeet and colleagues identified risk factors that lead towards software failure. The key four factors were immature technology, reliance on a few personal, lack of client support and lack of competence. Based on these four factors, a tool was developed to predict the delay and better management of time and cost in software project development. In addition, Faliagka and colleagues [15] proposed an online job recommender system for recruiting better human resource based on machine learning algorithms. The tool collects the candidate's basic information from his/her LinkedIn profile and reputation from his/her social presence and activities. Based on these two parameters, the algorithm compares the prerequisites of job requirements, calculates candidates' final scores and recommends the high score candidate.

4.4 RQ4. What are the limitations of the existing research studies?

After extensively analyzing the literature about software influencing factors, various limitations were found in existing research studies, which are summarized in the investigation report (attached in appendix-1). Multiple studies have suggested that a survey from the software industry is needed to identify reasons for software project failure. Various sub-items for cost and time estimation, scope creep, human resource and risk assessment should also be explored for better understanding and planning. There is a need to improve and expand the existing algorithms to cover other dimensions than just time and cost estimation. In addition, some of the proposed algorithms need some improvements by integrating other components of project estimation and management. In addition, majority of the current algorithms do not consider knowledge from previously developed software projects. Each study along with its methodology and limitations is discussed in the investigation report (attached in appendix-I)

5. Research Contribution and Recommendation

Software projects fail for so many diverse reasons which are difficult to be lumped into one category as illustrated by this research effort. To overcome this issue, this research proposes a framework for software project failure factors as shown in Figure 8. The framework consolidates current state of the art by analyzing existing studies, algorithms, automation tools and suggested failure factors. The framework also lumps various failure factors into three main categories based on their presence in literature which might help in the management of software projects. It also highlights the research gap in this domain as most of the algorithms and automation tools only support the estimation phase. In brief, this paper can work as a coherent source of literature on the failure of software projects, readily available for both researchers and practitioners.

Most of the studies explored that expert judgment is a primary approach for estimation of time and cost. Therefore, inexperienced project managers wrongly estimating time and cost are the primary cause of software project failure. Wrong estimation highly influences others factors in terms of task completion. Based on our findings from existing literature, following recommendations may help in building high confidence level for managing projects successfully.

(1) Training and usage of standard software project management tools can be the key to avoid failure factors.
(2) Experienced project managers should always be involved during the estimation process to minimize the influence of other identified factors on software project failure.
(3) Project time and cost are identified as the most contributing factors of most software projects failures, which should be emphasized more in planning a software project.
(4) Project managers should focus on good governance, risk management and regulatory factors to stay ahead of the competition.

6. Conclusion and Future Work

This study explored the literature related to software project failure to highlight its influencing factors. It also aimed to identify current trends in this field and gaps present in the existing research. Total of 2,171 studies

were observed with the combination of manual snowballing and automated searches, out of which 68 studies were further explored in detail according to the adapted investigation protocol. From these 68 selected studies, 13 influencing factors were identified and classified into three main categories. The four main influencing factors are wrong estimation of time and cost, scope creeping, inadequate human resource and risk control. The analysis further reflected that wrong estimation of time and cost is the major factor causing software project failures. Along with these major factors, five significant factors were also identified, which include communication issues, lack of planning, lack of user involvement, unrealistic expectations and lack of executive support. In addition, some insignificant factors were also identified which are lack of resources, technology illiteracy, staff not rewarded trust and knowledge sharing. The most influencing factor identified in this research work is time and cost estimation. The percentage of risk towards project failures can be greatly reduced if time and cost estimation is adequately evaluated.

Apart from factor identification, this study also explored various methodologies adopted in existing research. Three major adapted methodologies related to software project failure are systematic literature review (47.06%), empirical study (35.29%) and case studies (17.65%). Further, this study identified existing algorithms and automation tools in the domain and found that only 13.23% research studies proposed some algorithm to resolve the issue of software project failure and only 4.41% research studies developed some tool to aid the situation. In the future, an automated tool will be developed for better decision making for the scope management, resource selection, allocation and risk management along with time and cost estimation. A complete automation solution that can help with all four major failure factors will also be developed. Overall results indicated that more emphasis towards major identified factors can help in software project management and also put a positive impact on the successful completion of software projects. Successfully completion of software projects positively impact the software exports of a country, which in turn not only enhance the economy but also creates new jobs.

References

[1] K. Conboy, "Project failure en masse: a study of loose budgetary control in ISD projects," European Journal of Information Systems, vol. 19, pp. 273-287, 2010.

[2] A. Bharadwaj, M. Keil, and M. Mähring, "Effects of information technology failures on the market value of firms," The Journal of Strategic Information Systems, vol. 18, pp. 66-79, 2009.

[3] R. Stanley and L. Uden, "Why projects fail, from the perspective of service science," in 7th international conference on knowledge management in organizations: service and cloud computing, 2013, pp. 421-429.

[4] S. Dhir, D. Kumar, and V. Singh, "Success and Failure Factors that Impact on Project Implementation using Agile Software Development Methodology," in Software Engineering, ed: Springer, 2019, pp. 647-654.

[5] F. A. Batarseh and A. J. Gonzalez, "Predicting failures in agile software development through data analytics," Software Quality Journal, vol. 26, pp. 49-66, 2018.

[6] S. M. Sarif, S. Ramly, R. Yusof, N. A. A. Fadzillah, and N. Y. bin Sulaiman, "Investigation of Success and Failure Factors in IT Project Management," in International Conference on Kansei Engineering & Emotion Research, 2018, pp. 671-682.

[7] D. L. Hughes, Y. K. Dwivedi, A. C. Simintiras, and N. P. Rana, "Project failure and its contributing factors," in Success and failure of IS/IT projects, ed: Springer, 2016, pp. 3-25.

[8] R. Popli and N. Chauhan, "Cost and effort estimation in agile software development," in Optimization, Reliabilty, and Information Technology (ICROIT), 2014 International Conference on, 2014, pp. 57-61.

[9] Z. Jalil and A. Hanif, "Improving management of outsourced software projects in Pakistan," in Computer Science and Information Technology, 2009. ICCSIT 2009. 2nd IEEE International Conference on, 2009, pp. 524-528.

[10] I. ul Hassan, N. Ahmad, and B. Zuhaira, "Calculating completeness of software project scope definition," Information and Software Technology, vol. 94, pp. 208-233, 2018.

[11] K. L. Ramage, "Scope Management Strategies for Engineering Leaders to Improve Project Success Rates," 2018.

[12] S. Amjad, N. Ahmad, T. Saba, A. Anjum, U. Manzoor, M. A. Balubaid, et al., "Calculating completeness of agile scope in scaled agile development," IEEE Access, vol. 6, pp. 5822-5847, 2018.

[13] T. Clancy, "The Standish Group Report CHAOS," Project Smart, available at: www. projectsmart. co. uk/white-papers/chaos-report. pdf (accessed March 7, 2016).[Google Scholar] 2014.

[14] F. S. Gharehchopogh, I. Maleki, and S. R. Khaze, "A Novel Particle Swarm Optimization Approach For Software Effort Estimation," International Journal of Academic Research, vol. 6, 2014.

[15] E. Faliagka, L. Iliadis, I. Karydis, M. Rigou, S. Sioutas, A. Tsakalidis, et al., "On-line consistent ranking on e-recruitment: seeking the truth behind a well-formed CV," Artificial Intelligence Review, vol. 42, pp. 515-528, 2014.

[16] G. S. Walia and J. C. Carver, "A systematic literature review to identify and classify software requirement errors," Information and Software Technology, vol. 51, pp. 1087-1109, 2009.

[17] F. S. Gharehchopogh, I. Maleki, and S. R. Khaze, "A novel particle swarm optimization approach for software effort estimation," International Journal of Academic Research, Part A, vol. 6, pp. 69-76, 2014.

[18] M. Jørgensen, "The Use of Precision of Software Development Effort Estimates to Communicate Uncertainty," in International Conference on Software Quality, 2016, pp. 156-168.

[19] J. Ashraf, N. S. Khattak, and A. M. Zaidi, "Why do public sector IT projects fail," in Informatics and Systems (INFOS), 2010 The 7th International Conference on, 2010, pp. 1-6.
[20] W. Al-Ahmad, K. Al-Fagih, K. Khanfar, K. Alsamara, S. Abuleil, and H. Abu-Salem, "A taxonomy of an IT project failure: root causes," International Management Review, vol. 5, pp. 93-104, 2009.
[21] S. Kujala, "User involvement: a review of the benefits and challenges," Behaviour & information technology, vol. 22, pp. 1-16, 2003.
[22] A. Hussain, E. O. Mkpojiogu, and F. M. Kamal, "The role of requirements in the success or failure of software projects," International Review of Management and Marketing, vol. 6, 2016.
[23] J. Luftman, K. Lyytinen, and T. ben Zvi, "Enhancing the measurement of information technology (IT) business alignment and its influence on company performance," Journal of Information Technology, vol. 32, pp. 26-46, 2017.
[24] I. Thamarai and S. Murugavalli, "Model for improving the accuracy of relevant project selection in analogy using differential evolution algorithm," Sādhanā, vol. 42, pp. 23-31, 2017.
[25] C. General, "Contracting for Computer Software Development," General Accounting Office report FGMSD-80, vol. 4, 1979.
[26] "The Standish Group chaos report," 1994.
[27] M. Jorgensen and M. Shepperd, "A systematic review of software development cost estimation studies," IEEE Transactions on software engineering, vol. 33, 2007.
[28] K. Dikert, M. Paasivaara, and C. Lassenius, "Challenges and success factors for large-scale agile transformations: A systematic literature review," Journal of Systems and Software, vol. 119, pp. 87-108, 2016.
[29] A. Idri, M. Hosni, and A. Abran, "Systematic literature review of ensemble effort estimation," Journal of Systems and Software, vol. 118, pp. 151-175, 2016.
[30] G. Guillaume-Joseph, "Improving Software Project Outcomes through Predictive Analytics," The George Washington University, 2016.
[31] E. Hossain, M. A. Babar, and H.-y. Paik, "Using scrum in global software development: a systematic literature review," in Global Software Engineering, 2009. ICGSE 2009. Fourth IEEE International Conference on, 2009, pp. 175-184.
[32] I. Inayat, S. S. Salim, S. Marczak, M. Daneva, and S. Shamshirband, "A systematic literature review on agile requirements engineering practices and challenges," Computers in human behavior, vol. 51, pp. 915-929, 2015.
[33] R. K. Gupta, P. Manikreddy, and K. Arya, "Pragmatic Scrum Transformation: Challenges, Practices & Impacts During the Journey A case study in a multi-location legacy software product development team," in Proceedings of the 10th Innovations in Software Engineering Conference, 2017, pp. 147-156.
[34] N. Cerpa, M. Bardeen, B. Kitchenham, and J. Verner, "Evaluating logistic regression models to estimate software project outcomes," Information and Software Technology, vol. 52, pp. 934-944, 2010.
[35] S. Komchaliaw and P. Wongthongtham, "A state of the art review on software project performance management," in Digital Ecosystems and Technologies (DEST), 2010 4th IEEE International Conference on, 2010, pp. 653-655.
[36] I. Attarzadeh and S. H. Ow, "Project management practices: the criteria for success or failure," Communications of the IBIMA, vol. 1, pp. 234-241, 2008.
[37] V. Damasiotis, P. Fitsilis, P. Considine, and J. O'Kane, "Analysis of software project complexity factors," in Proceedings of the 2017 International Conference on Management Engineering, Software Engineering and Service Sciences, 2017, pp. 54-58.
[38] B. Kitchenham, D. Budgen, and P. Brereton, "Evidence-Based Software Engineering, Empirical SE, Software Design," ed: CRC Press, Boca Raton, FL, 2015.
[39] P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner, and M. Khalil, "Lessons from applying the systematic literature review process within the software engineering domain," Journal of systems and software, vol. 80, pp. 571-583, 2007.
[40] V. R. Basili and D. M. Weiss, "A methodology for collecting valid software engineering data," IEEE Transactions on software engineering, pp. 728-738, 1984.
[41] H. Zhang, M. A. Babar, and P. Tell, "Identifying relevant studies in software engineering," Information and Software Technology, vol. 53, pp. 625-637, 2011.
[42] J. G. Borade and V. R. Khalkar, "Software project effort and cost estimation techniques," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 3, 2013.
[43] C. Ebert and J. De Man, "Requirements uncertainty: influencing factors and concrete improvements," in Proceedings of the 27th international conference on Software engineering, 2005, pp. 553-560.
[44] J. D. Musa, Software reliability engineering: more reliable software, faster and cheaper: Tata McGraw-Hill Education, 2004.
[45] N. Azhar, R. U. Farooqui, and S. M. Ahmed, "Cost overrun factors in construction industry of Pakistan," in First International Conference on Construction In Developing Countries (ICCIDC–I), Advancing and Integrating Construction Education, Research & Practice, 2008, pp. 499-508.
[46] K. Jeet, N. Bhatia, and R. S. Minhas, "A model for estimating the impact of low productivity on the schedule of a software development project," ACM SIGSOFT Software Engineering Notes, vol. 36, pp. 1-6, 2011.
[47] J. Verner, J. Sampson, and N. Cerpa, "What factors lead to software project failure?," in Research Challenges in Information Science, 2008. RCIS 2008. Second International Conference on, 2008, pp. 71-80.
[48] R. Pham, S. Kiesling, L. Singer, and K. Schneider, "Onboarding inexperienced developers: struggles and perceptions regarding automated testing," Software Quality Journal, pp. 1-30, 2016.
[49] A. Trendowicz and R. Jeffery, "Software Project Effort Estimation," Foundations and Best Practice Guidelines for Success, Constructive Cost Model–COCOMO pags, pp. 277-293, 2014.
[50] T. M. Connolly, E. A. Boyle, E. MacArthur, T. Hainey, and J. M. Boyle, "A systematic literature review of empirical

evidence on computer games and serious games," Computers & Education, vol. 59, pp. 661-686, 2012.
[51] B. Kitchenham, L. Pickard, and S. L. Pfleeger, "Case studies for method and tool evaluation," IEEE software, vol. 12, pp. 52-62, 1995.
[52] S. Weibelzahl and G. Weber, "Advantages, opportunities and limits of empirical evaluations: Evaluating adaptive systems," KI, vol. 16, pp. 17-20, 2002.
[53] A. Bharadwaj, "Empirical Methodology and Findings," in Environmental Regulations and Innovation in Advanced Automobile Technologies, ed: Springer, 2018, pp. 81-99.
[54] B. Flyvbjerg, "Five misunderstandings about case-study research," Qualitative inquiry, vol. 12, pp. 219-245, 2006.
[55] C. Andersson and P. Runeson, "A spiral process model for case studies on software quality monitoring—method and metrics," Software Process: Improvement and Practice, vol. 12, pp. 125-140, 2007.
[56] C. Stylianou and A. S. Andreou, "Intelligent software project scheduling and team staffing with genetic algorithms," in Artificial Intelligence Applications and Innovations, ed: Springer, 2011, pp. 169-178.
[57] S. Bahlerao and M. Ingle, "Generalized Agile Estimation Method," International Journal on Advanced Science, Engineering and Information Technology, vol. 1, pp. 262-267, 2011.
[58] A. Bia, R. Muñoz, and J. Gómez, "Estimating digitization costs in digital libraries using DiCoMo," in International Conference on Theory and Practice of Digital Libraries, 2010, pp. 136-147.
[59] B. Boehm, B. Clark, E. Horowitz, C. Westland, R. Madachy, and R. Selby, "Cost models for future software life cycle processes: COCOMO 2.0," Annals of software engineering, vol. 1, pp. 57-94, 1995.
[60] J. E. Matson, B. E. Barrett, and J. M. Mellichamp, "Software development cost estimation using function points," IEEE Transactions on Software Engineering, vol. 20, pp. 275-287, 1994.
[61] B. W. Boehm, R. Madachy, and B. Steece, Software cost estimation with Cocomo II with Cdrom: Prentice Hall PTR, 2000.
[62] F. Reyes, N. Cerpa, A. Candia-Véjar, and M. Bardeen, "The optimization of success probability for software projects using genetic algorithms," Journal of Systems and Software, vol. 84, pp. 775-785, 2011.
[63] P. Shoval and O. Feldman, "A combination of the Mk-II Function Points software estimation method with the ADISSA methodology for systems analysis and design," Information and Software Technology, vol. 39, pp. 855-865, 1997.
[64] C. R. Symons, Software sizing and estimating: Mk II FPA (function point analysis): John Wiley & Sons, Inc., 1991.
[65] P. Shoval, "ADISSA: Architectural design of information systems based on structured analysis," Information systems, vol. 13, pp. 193-210, 1988.
[66] M. Adnan and M. Afzal, "Ontology Based Multiagent Effort Estimation System for Scrum Agile Method," IEEE Access, vol. 5, pp. 25993-26005, 2017.
[67] T. Moh'd MI, A.-T. Haroon, and A. Elsheikh, "Software development projects: An investigation into the factors that affect software project success/failure in Jordanian firms," in Applications of Digital Information and Web Technologies, 2008. ICADIWT 2008. First International Conference on the, 2008, pp. 246-251.
[68] N. Cerpa and J. M. Verner, "Why did your project fail?," Communications of the ACM, vol. 52, pp. 130-134, 2009.
[69] J. Coelho and M. T. Valente, "Why modern open source projects fail," in Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, 2017, pp. 186-196.
[70] R. Berntsson-Svensson and A. Aurum, "Successful software project and products: An empirical investigation," in Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering, 2006, pp. 144-153.
[71] S. Dalal and R. S. Chhillar, "Empirical study of root cause analysis of software failure," ACM SIGSOFT Software Engineering Notes, vol. 38, pp. 1-7, 2013.
[72] D. Stankovic, V. Nikolic, M. Djordjevic, and D.-B. Cao, "A survey study of critical success factors in agile software projects in former Yugoslavia IT companies," Journal of Systems and Software, vol. 86, pp. 1663-1678, 2013.
[73] M. Jørgensen, "Failure factors of small software projects at a global outsourcing marketplace," Journal of systems and software, vol. 92, pp. 157-169, 2014.
[74] K. A. Demir, "A Survey on Challenges of Software Project Management," Software Engineering Research and Practice, vol. 2009, pp. 579-585, 2009.
[75] J. Hihn and H. Habib-agahi, "Cost estimation of software intensive projects: A survey of current practices," in Proceedings of the 13th international conference on Software engineering, 1991, pp. 276-287.
[76] Z. Siting, H. Wenxing, Z. Ning, and Y. Fan, "Job recommender systems: a survey," in Computer Science & Education (ICCSE), 2012 7th International Conference on, 2012, pp. 920-924.
[77] K. Molokken and M. Jorgensen, "A review of software surveys on software effort estimation," in Empirical Software Engineering, 2003. ISESE 2003. Proceedings. 2003 International Symposium on, 2003, pp. 223-230.
[78] F. S. Butt, M. Liaqat, M. Khan, W. Nisar, and E. U. Munir, "Common Factors in the Successful Software Projects in Pakistan‘s Software Industry," World Applied Sciences Journal, vol. 23, pp. 1176-1185, 2013.
[79] A. Arshad, "A Survey on Software Cost Estimation in the Pakistani Software Industry," IJCER, vol. 3, pp. 13-19, 2014.
[80] J. M. Verner, W. M. Evanco, and N. Cerpa, "State of the practice: An exploratory analysis of schedule estimation and software project success prediction," Information and Software Technology, vol. 49, pp. 181-193, 2007.
[81] M. Jørgensen, "A survey on the characteristics of projects with success in delivering client benefits," Information and Software Technology, vol. 78, pp. 83-94, 2016.
[82] T. Chow and D.-B. Cao, "A survey study of critical success factors in agile software projects," Journal of systems and software, vol. 81, pp. 961-971, 2008.
[83] V. Lenarduzzi and D. Taibi, "Can Functional Size Measures Improve Effort Estimation in SCRUM?," in ICSEA-International Conference on Software Engineering and Advances, Nice, France, 2014.

[84] S. Falahah, "Risk management assessment using SERIM method," in Int. Conf. on e-Education, Entertainment and e-Management, 2011, pp. 340-343.
[85] R. L. Glass, "IT Failure Rates-70% or 10-15%?," IEEE Software, vol. 22, 2005.
[86] J. L. Eveleens and C. Verhoef, "The rise and fall of the chaos report figures," IEEE software, vol. 27, pp. 30-36, 2010.
[87] M. Warkentin, R. S. Moore, E. Bekkering, and A. C. Johnston, "Analysis of systems development project risks: An integrative framework," ACM SIGMIS Database, vol. 40, pp. 8-27, 2009.
[88] D. Damian, J. Chisan, L. Vaidyanathasamy, and Y. Pal, "Requirements engineering and downstream software development: Findings from a case study," Empirical Software Engineering, vol. 10, pp. 255-283, 2005.
[89] S. Parthasarathy and M. Daneva, "An approach to estimation of degree of customization for ERP projects using prioritized requirements," Journal of systems and software, vol. 117, pp. 471-487, 2016.
[90] C. Wohlin and A. A. Andrews, "Prioritizing and assessing software project success factors and project characteristics using subjective data," Empirical Software Engineering, vol. 8, pp. 285-308, 2003.
[91] J. Keung, R. Jeffery, and B. Kitchenham, "The challenge of introducing a new software cost estimation technology into a small software organisation," in Software Engineering Conference, 2004. Proceedings. 2004 Australian, 2004, pp. 52-59.
[92] J. Debari, O. Mizuno, T. Kikuno, N. Kikuchi, and M. Hirayama, "On deriving actions for improving cost overrun by applying association rule mining to industrial project repository," in International Conference on Software Process, 2008, pp. 51-62.
[93] A. A. Andrews, P. Beaver, and J. Lucente, "Towards better help desk planning: Predicting incidents and required effort," Journal of Systems and Software, vol. 117, pp. 426-449, 2016.
[94] S. Dragicevic, S. Celar, and M. Turic, "Bayesian network model for task effort estimation in agile software development," Journal of Systems and Software, vol. 127, pp. 109-119, 2017.
[95] M. Jin, "The Development of the Chinese ICT Industry and Japanese Firms' Offshoring: With a Focus on Dalian's Case," in Innovative ICT Industrial Architecture in East Asia, ed: Springer, 2017, pp. 99-114.
[96] A. B. Nassif, D. Ho, and L. F. Capretz, "Towards an early software estimation using log-linear regression and a multilayer perceptron model," Journal of Systems and Software, vol. 86, pp. 144-160, 2013.
[97] M. Soualhia, F. Khomh, and S. Tahar, "Task Scheduling in Big Data Platforms: A Systematic Literature Review," Journal of Systems and Software, vol. 134, pp. 170-189, 2017.
[98] B. B. Chua, D. V. Bernardo, and J. Verner, "Criteria for estimating effort for requirements changes," in European Conference on Software Process Improvement, 2008, pp. 36-46.
[99] D. Dalcher, "Rethinking success in software projects: looking beyond the failure factors," in Software project management in a changing world, ed: Springer, 2014, pp. 27-49.
[100] A. Seetharaman, M. Senthilvelmurugan, and T. Subramanian, "Budgeting and accounting of software cost: Part 1," Journal of Digital Asset Management, vol. 1, pp. 347-359, 2005.
[101] F. González-Ladrón-de-Guevara, M. Fernández-Diego, and C. Lokan, "The usage of ISBSG data fields in software effort estimation: A systematic mapping study," Journal of Systems and Software, vol. 113, pp. 188-215, 2016.
[102] M. Keil, J. Mann, and A. Rai, "Why software projects escalate: An empirical analysis and test of four theoretical models," Mis Quarterly, pp. 631-664, 2000.
[103] K. V. Vasantrao, "Understanding need of flexible software development approach using "The Economic Model"," in Electronics Computer Technology (ICECT), 2011 3rd International Conference on, 2011, pp. 240-243.
[104] Z. Nie, X.-h. Jiang, J.-c. Liu, and H. Yang, "Completion Time Estimation for Instances of Generalized Well-Formed Workflow," in Parallel and Distributed Processing with Applications, 2009 IEEE International Symposium on, 2009, pp. 611-616.
[105] R. Ashman, "Project estimation: a simple use-case-based model," IT professional, vol. 6, pp. 40-44, 2004.
[106] M. Jørgensen, "Top-down and bottom-up expert estimation of software development effort," Information and Software Technology, vol. 46, pp. 3-16, 2004.
[107] M. Usman, E. Mendes, F. Weidt, and R. Britto, "Effort estimation in agile software development: a systematic literature review," in Proceedings of the 10th International Conference on Predictive Models in Software Engineering, 2014, pp. 82-91.
[108] R. Popli and N. Chauhan, "Agile estimation using people and project related factors," in Computing for Sustainable Global Development (INDIACom), 2014 International Conference on, 2014, pp. 564-569.
[109] Y. K. Dwivedi, K. Ravichandran, M. D. Williams, S. Miller, B. Lal, G. V. Antony, et al., "IS/IT project failures: a review of the extant literature for deriving a taxonomy of failure factors," in International Working Conference on Transfer and Diffusion of IT, 2013, pp. 73-88.
[110] M. Jørgensen and K. Moløkken-Østvold, "How large are software cost overruns? A review of the 1994 CHAOS report," Information and Software Technology, vol. 48, pp. 297-301, 2006.
[111] Pareto, V., 1971. Manual of Political Economy. Translation of 1906 Edition, Augustus M. Kelley, New York.

Appendix-1

Ref No	Methodology	Algorithm	Automation	Limitation
[47]	Empirical study	X	X	The survey data was self-reported and needed more empirical study implication to investigate the issue.
[67]	Empirical study	X	X	The survey data was self-reported to only Jordanian software Firms.
[43]	Empirical study	X	X	More study that is an empirical needed for proper extraction and management of requirements.
[9]	Empirical study	X	X	As per review, need a tool to handle scheduling issues.
[68]	Empirical Study	X	X	Need to improve the estimation process for prediction of a software project.
[69]	Empirical study	X	X	The author recommended further investigation into proactive strategies to avoid the failure of software projects.
[70]	Empirical study	X	X	The empirical study results showed that there is a need for a tool for better estimation of time and cost.
[71]	Empirical study	X	X	The author suggested that new integrated software testing technique needs to be developed for prediction of software projects.
[34]	Empirical study	X	X	The empirical study results showed that the tool of estimation could minimize the software failure issues.
[72]	Empirical study	X	X	The author recommended more empirical study need and developed a tool that handles the estimation and management tasks.
[73]	Empirical study	X	X	The author suggested there was a need for effective estimation process for handling large-scale agile projects.
[74]	Empirical study	X	X	More empirical studies required to identify the factors influencing the software project failure.
[36]	Empirical study	X	X	Needed to save the knowledge and use in future projects for proper project estimation and management
[13]	Empirical study	X	X	More empirical studies needed to identify the influencing reasons behind software project failure.
[75]	Empirical study	X	X	The author recommended more empirical study need and developed an algorithm to handle the software estimation issues.
[76]	Empirical study	X	X	The author suggested that there is a need to generate the enrich user profile for accurate selection of human resource.
[77]	Empirical study	X	X	Lack of surveys is there, like a comprehensive evaluation of the logic for effort and schedule overruns.
[78]	Empirical study	X	X	The results of the empirical study showed that data on similar projects not available that used for estimation of upcoming software projects.
[79]	Empirical study	X	X	The author recommended that more empirical study is needed in identifying the causes of software project failure.
[80]	Empirical study	X	X	Improve estimation and management process to avoid the failure.
[81]	Empirical study	X	X	Project management estimation and control need to investigate further.
[82]	Empirical study	X	X	The more empirical study further need to identify the failure causes.
[83]	Empirical study	X	X	Future work includes the replication of this study in an industrial context with an understandable scrum process.
[66]	Empirical study	✓	✓	In future, an approach will be needed that handle distributed agile based project knowledge as well as to mitigate the coordination and communication challenges of geographically distributed development teams.
[84]	Case Study	X	X	The author recommended that further investigating the wrong estimation issues to overcome the software failure.
[19]	Case Study	X	X	There is a need for Identifying the effective way to handle requirements and schedule overrun.
[85]	Case Study	X	X	There is a need an approach that improves the estimation process.
[86]	Case Study	X	X	The author suggests that there is need to improve the estimation process for cost and time estimation.
[87]	Case Study	X	X	The author recommended in future there is need an efficient managerial developments and adequate allocation of human resources.
[88]	Case Study	X	X	The author suggested culture change in organizations and requirement engineering process further need to investigate.
[57]	Case studies	✓	X	In future, needed to mature the existing algorithm for estimation of time and cost.
[89]	Case study	X	X	The limitation of case study showed that there is need to save the learning and used for estimation in updating software projects
[90]	Case study	X	X	The author suggested that in future there is need to use previously developed software project data for upcoming software projects.
[14]	Case study	X	X	More case study is needed to investigate the estimation issues.
[91]	Case study	X	X	More work is needed in the domain of early prediction in future.
[92]	Case study	X	X	Further need to compare that algorithm with others studies.
[42]	Systemic Literature review	X	X	The author recommends a technique required to save the lesson of current or past developed software project data.
[30]	Systemic Literature review	X	X	There is need an approach to use the knowledge of developed software projects for future estimation of time and cost.
[93]	Systemic Literature review	X	X	Need to save and use previous software project data for new software handling and estimation.
[29]	Systematic literature review	X	X	Knowledge of developed software projects needed to save for the future and used for upcoming software projects.

[94]	Systematic literature review	X	X	The future research needed to save the data in knowledge based and used in future.
[95]	Systematic literature review	X	X	Data needed for similar software projects for estimation of upcoming software projects.
[96]	Systematic literature review	X	X	Predict new software projects based on already developing software project data. However, needed of similar types of software projects data.
[62]	Systematic literature review	✓	X	Expand the existing algorithms and integrate other components for different software predication.
[63]	Systematic literature review	✓	X	Needed to save previously developed software project data for estimation of upcoming software projects.
[97]	Systematic Literature Review	X	X	The author suggested that future research need to design and manage the task and jobs scheduling in Hadoop.
[98]	Systematic Literature Review	X	X	Further empirical study needed to check the existing framework for estimation.
[99]	Systematic Literature Review	X	X	Need to develop a guideline for requirement management.
[56]	Systematic Literature Review	✓	X	Needed to improve the existing algorithm for time and cost estimation.
[58]	Systematic Literature Review	✓	X	The algorithm may need to improve based on the previously developed software data
[24]	Systematic Literature Review	✓	X	More empirical data will required to test the developed algorithm.
[100]	Systematic Literature Review	X	X	In the future model will be developed for better estimation of time and cost.
[28]	Systematic Literature Review	X	X	The author suggested more study needed to identify the influencing factors in further.
[101]	Systematic Literature Review	X	X	More variable was needed to in-depth discuss for effort estimation driver over time.
[102]	Systematic Literature Review	X	X	The literature showed that more dataset would be needed for upcoming software projects.
[103]	Systematic Literature Review	X	X	Same data of previously developed software projects was a need for future estimation of software projects.
[104]	Systematic Literature Review	X	X	Same data of previously developed software projects was a need for future estimation of software projects.
[105]	Systematic Literature Review	X	X	A dataset of same software projects was needed for estimation and decision making of upcoming software projects.
[15]	Systematic Literature Review	✓	✓	In future, more attribute accessed from personality for accurate selection of human resource.
[106]	Systematic Literature Review	X	X	The author recommended saving previously developed software project data for upcoming software projects.
[93]	Systematic Literature Review	X	X	In further need historical data for upcoming software projects estimation.
[107]	Systematic Literature Review	X	X	The author strongly recommended investigating the estimation studies in scrum and XP context.
[108]	Systematic Literature Review	✓	X	In future, save the historical data for upcoming software projects.
[35]	Systemic Literature review	X	X	Trust sharing measurement need to investigate further.
[46]	Systemic Literature review	✓	✓	Some more factors are needed to study that impact on the productivity or schedule overrun.
[37]	Systemic Literature review	X	X	The literature results showed that project management tools would need to avoid software failure.
[109]	Systemic Literature review	X	X	The knowledge management tools would require avoiding the software project failure.
[110]	Systemic Literature review	X	X	The literature results showed that an effective technique would need in future for time and cost estimation.