Computers, Materials & Continua
DOI: 10.32604/cmc.2023.030162
Article

Tech Science Press

Empirical Analysis of Software Success Rate Forecasting During Requirement Engineering Processes

Muhammad Hasnain[1], Imran Ghani[2], Seung Ryul Jeong[3,*], Muhammad Fermi Pasha[4], Sardar Usman[5] and Anjum Abbas[6]

[1]Lahore Leads University, Lahore, 42000, Pakistan
[2]Computer and Information Sciences, Virginia Military Institute, Lexington, VA, 24450, USA
[3]Graduate School of Business IT, Kookmin University, Seoul, 136, Korea
[4]School of Information Technology, Monash University, Bandar Sunway, 47500, Malaysia
[5]Department of Computer Science and Software Engineering, Grand Asian University Sialkot, 51040, Pakistan
[6]Government Sarwar Shaheed Associate College, Gujjar Khan, Rawalpindi, 46000, Pakistan
*Corresponding Author: Seung Ryul Jeong. Email: srjeong@kookmin.ac.kr
Received: 19 March 2022; Accepted: 13 June 2022

Abstract: Forecasting on success or failure of software has become an interesting and, in fact, an essential task in the software development industry. In order to explore the latest data on successes and failures, this research focused on certain questions such as is early phase of the software development life cycle better than later phases in predicting software success and avoiding high rework? What human factors contribute to success or failure of a software? What software practices are used by the industry practitioners to achieve high quality of software in their day-to-day work? In order to conduct this empirical analysis a total of 104 practitioners were recruited to determine how human factors, misinterpretation, and miscommunication of requirements and decision-making processes play their roles in software success forecasting. We discussed a potential relationship between forecasting of software success or failure and the development processes. We noticed that experienced participants had more confidence in their practices and responded to the questionnaire in this empirical study, and they were more likely to rate software success forecasting linking to the development processes. Our analysis also shows that cognitive bias is the central human factor that negatively affects forecasting of software success rate. The results of this empirical study also validated that requirements' misinterpretation and miscommunication were the main causes behind software systems' failure. It has been seen that reliable, relevant, and trustworthy sources of information help in decision-making to predict software systems' success in the software industry. This empirical study highlights a need for other software practitioners to avoid such bias while working on software projects. Future investigation can be performed to identify the other human factors that may impact software systems' success.

Keywords: Cognitive bias; misinterpretation of requirements; miscommunication; software success and failure prediction; decision making

1 Introduction

The predictability of software success is one of the major concerns in the software industry. To predict or forecast software success or failure, one can use different software development methods and tools, commonly used in the software development industry [1]. A considerable body of knowledge on the software defect prediction links cost-saving and different phases of the "software development life cycle" (SDLC). Studies [2,3] explore software defect prediction (SDP) as a cost reduction solution during the software testing phase. Machine learning classifiers have been used in these studies to predict software defects.

A study [4] reveals that most of the outsourced software projects are affiliated with a high failure rate. While providing insights, the existing literature presents some conflicting results. Studies [5,6] find that software project managers' active involvement in schedule negotiation and adequate requirements' information are key factors for the success of software projects. Transformational leadership, and value congruence are key factors for the success of software [7]. A study [8] finds that software projects' success or failure rate is related to user satisfaction, requirement engineering, start-up pivots, and retrospective discussions. Study [9] results show that software metrics could be applied at different phases of the SDLC. Still, it is insignificant to predict software success or failure rate at lateral phases if faults or errors are not handled at an earlier phase. Our empirical study makes several contributions to software success or failure estimation during the software life cycle.

- First, this work is focused on the knowledge of software practitioners to know human factors' impacts on the software success prediction. We study cognitive bias as a human factor that can impact forecasting the success of software.
- Second, empirical analysis presents software practitioners' ratings to identify the software life cycle's accurate phases to forecast software systems' success. An accurate phase is meant to specify one or more software development phases for its success or failure prediction. The motivation behind the success or failure prediction is to save costs on rescheduling and redeveloping the work.
- Third, our study results and analysis validate that requirements' misinterpretation and miscommunication during the requirement engineering process can create conflicts and damage software success. We want to listen to the software practitioners from their rating; the causes mentioned above can decrease the software success in the later phases of the software life cycle.
- Fourth, our study presents an analysis of decision making from better information judgment. We analyze the empirical results to know whether accurate software success forecasting can help expand software specifications in the future for required changes.

Structure of the remaining paper is as follows:

Section 2 presents related work to our research topic, Section 3 presents the empirical method, Section 4 presents empirical results and findings, Section 5 presents limitations of this study, and Section 6 concludes the paper.

2 Related Works

In a study, Chotisarn and Prompoon [10] focused on the behavioral perspectives, particularly the psychological impacts on the cognitive bias during the early phase of software requirements gathering and specification processes. This study was a first attempt for forecasting the software damage rate at an early stage of requirement gathering. However, the sample population used for this study was

limited. The use of multiple regression and correlation analysis on variables resulted in software failure or success rate.

A multilayer perception model was proposed to solve the issue of software estimation. Researchers in [11] called the software requirement gathering a critical phase for software success estimation. The Use-case point (UCP) model was focused on the use-case diagram. This model outperformed the original UCP model in some respects when used to estimate small-sized projects.

High-quality systems could only be developed if a defect prevention strategy is adopted at all stages of the software development life cycle. The defect prevention strategy did not only reduce the cost but also the time as well. The concerned research study [12] was to implement a model-based strategy on the requirement gathering phase. Achieving high-quality software through the agile development of software resulted in several defect management methods, such as defect discovery, defect prevention, and resolution. Agile methodologies, based on the short iterations, helped developers overcome the defects.

A research effort was made on the software required for the space projects. The researchers' objective in [13] was to understand the most critical application domain's requirement issues. A well-known taxonomy, latterly used for thorough analysis, was applied. The results of the study revealed that there were 9.5 errors per hundred requirements. Most frequent errors were pointed out as external conflict/consistency, requirement completeness, and traceability. This work's effectiveness was limited only to classify the requirement problems but could not provide the solution to these requirement issues.

To cope with software failure, the study [14] devised the failure removal estimation model that considered the removal of failures occurred at the time interval. Reliability in this model was assured at the testing phase, considering test and debug time together. It was intended to ensure reliability from the early stage of testing to the lateral testing stage. A paired T-Test was applied to test the reliability between the proposed model and the existing models. Results showed the significant difference that was noted between the Okumoto model and the proposed model.

A recently published research [15] considers neural networks for time series forecasting to address complex issues regardless of any domain. Recurrent neural network (RNN) models can replace the univariate models used in production. However, many researchers are still reluctant to use them because they lack expert knowledge since it is not established in the forecasting community when to use deep learning models. Besides this, RNNs for large time-series dataset *vs.* short isolated datasets are still under research. It is suggested from the earlier mentioned research work that the research community can benefit from using both conventional forecasting techniques as well as the latest deep learning techniques if they precisely implement and apply them to real world issues It could be a better use of RNNs in software engineering processes if models are modified for an accurate prediction of software success. RNNs models are mostly applied to time series information and show limitations when applied to qualitative data.

Interaction failures, which arise either from the influence of software on hardware or the influence of hardware on the software, play a key role in a successful system's operations. It is reported that little work is undertaken to investigate the earlier mentioned interaction failures. To overcome this issue, the study [16] proposed a model for identifying a system's function from the requirements specification document. Identified functions were mapped to design components. The proposed model works as an early availability and reliability prediction model validated by the obtained availability and reliability values of a system.

Both open-source and software industries have witnessed continuous integration as a common practice. Software developers integrate the automated generated code to meet the changes frequently and quickly. However, the automated build process that generates code requires maintenance resulting in time and resource consumption. To predict the continuous integration outcome, the study [17] proposed a search based approach that aimed at finding the combination of continuous failure combination and their threshold values. Conflicting objectives, such as passed build and failed builds were dealt with by the proposed model. The proposed model outperformed the machine learning models by handling a better balance between two types of builds. However, machine learning is playing a key role in handling the classification and detection of malicious software code [18–20]. The conversion of malicious code into images is used to train and augment the classifiers.

Business and Information technology requirements have been modelled to derive automatically system requirements [21]. To develop a system according to business requirements and understand business process (BP) goals has become an important development in software engineering. It positively impacts the configuration of business as well as business requirements. Prior to this research a little works has been conducted on empirical research on requirement engineering. A recent review revealed that practitioners have less involvement in empirical research on RE topics [22]. Our study intends to bridge this gap by conducting an empirical research by involving industry practitioners. A qualitative study [23] has been undertaken to analyze the RE challenges and their solutions by implementing RE in software crowdsourcing. However, challenges such as software success or failure rate prediction remains unattended in the literature. Security feature of e-government projects are briefly investigated [24], success or failure of a project can be undertaken in future works. A Deep Fuzzy C-Mean (DFCM) technique creates new features from labelled and unlabeled data. Furthermore, data pre-processing on software performance prediction models is performed to ensure the quality assurance of software projects [25]. The proposed approach is mainly focused on classification and reduction of noisy information from extracted features.

Goal-oriented RE using the e-health business goals has been proposed and investigated regarding business and IT requirements [26]. Hence a health e-system is developed from modelling organization goals. The validity of approach is performed on emergency COVID-19 patients by checking them in hospitals. Although, system implementation is verified but its generalization to other e-health business goals is not performed. Thus, its success needs to be verified by using other business goals.

3 Research Methodology

Research methodology is widely used to obtain broad characterization regarding an important issue by collecting information from various information, including perceptions, opinions and attitudes [27]. An empirical analysis is well-suited strategy to collect information from large and targeted populations. Many surveying methods have been used in the literature and each method with its advantages and disadvantages. This study uses an online data collection method to quickly gather information, categorize and analyze the collected data. Another method in software engineering is interviewing that is usually more effort intensive [28]. Therefore, we have chosen online data collection method that is relatively easier for a software engineer to fill it. A framework has been proposed to support the research ideas in this study as shown in the following Fig. 1.

Fig. 1 shows us the main components of the research framework to support the proposed ideas in this study. All these components have been used to conduct the empirical research about linkage between the software success and software development processes in software development industry.

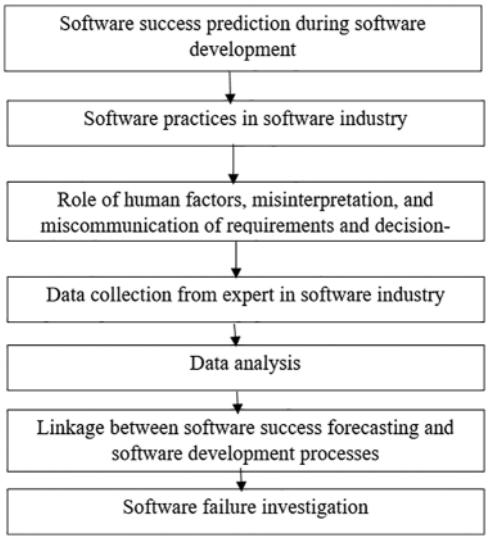

Figure 1: A research framework to support the ideas in this study

3.1 High-Level Goals and Research Question

Our empirical study uses the high-level goals and research questions. The goal is to understand the software practitioners' concerns about the software success rate estimation in domain of software industry. Based on this goal, we have raised the following research questions (RQs) with their motivations.

Tab. 1 shows the proposed research questions, their aspects from the empirical study's design. We could not present the entire empirical analysis because of space constraints in this paper. The final research question consisted of five sections. The first section was focused on gathering participants' profiles, and rest of four sections correspond to the RQs as shown in Tab. 2.

Table 1: List of proposed research questions, and their aspects from empirical study design

Research question	Aspect	Survey question
Practitioners	Profiles	i. Please specify your designation ii. Please indicate your experience in years.

(Continued)

Table 1: Continued

Research question	Aspect	Survey question
RQ1: How do human factors impact the software success rate forecasting?	Human factors	(1). Is software accuracy highly derived from precision in software success/failure rate estimation? (4). Is software damage rate estimation from human factors reliable forecasting? (6). Do cognitive concerns of software impact the human judgment about software damage rate forecasting? (7). Does a correlation between cognitive bias and software damage rate exit? (8). Does cognitive-based software estimation increase the efficiency of software damage rate estimation?
RQ2: Do software practitioners consider the requirement gathering phase as a vehicle to forecast the design or coding errors?	Recognizing the accurate phase	(9). Does the requirement gathering phase provide accurate software damage rate estimation? (10). Is it right that software damage rate can be determined through software requirement specification? (11). Is software failure/success forecasting more reliable in the early phases of software development?
RQ3: Do practitioners consider misinterpretation of requirements, and communication errors are the main causes behind software failure?	Requirements' misinterpretation, and communication errors	(12). Does the misinterpretation of requirements lead to software failure? (15). Do requirement changes result from poor communication between system stakeholders? (16). Does Poor communication between client and requirement engineer impact the software estimation? (17). Does poor communication during requirement engineering processes produce poor quality requirements? (18). Does communication error produce the delayed software requirement specification?

(Continued)

Table 1: Continued

Research question	Aspect	Survey question
RQ4: What are the means used by practitioners to predict the success or failure of software systems?	Decision making	(2). Does measurement of efforts for self-descriptiveness support the software success or failure rate estimation? (3). Does better judgment always reveal accurate forecasting of software failure rate? (5). Does software damage rate estimation provide the right direction for requirements' expand-ability in the future? (13). Do you think that decision-makers apply the information judgment to ensure the better software damage rate forecasting? (14). Is damage rate forecasting implemented to avoid the rework on software applications?

Table 2: Mapping of the original empirical study's questions to the research questions

Research questions and goals	Motivation
High-Level goal: To what extent software practitioners concern themselves with software success rate estimation?	To seek software practitioners' answers and interpret them for software success, rate estimation is the main motivation behind the research goal.
RQ1. How do human factors impact the software success rate forecasting?	Q1, Q4, Q6, Q7, and Q8 are questions about the impacts of human factors such as cognitive bias on the software success rate prediction.
RQ2. Do software practitioners consider the requirement gathering phase as a vehicle to forecast the design or coding errors?	Q9, Q10, and Q11 direct questions to determine whether the requirement engineering phase is the most appropriate (SDLC) phase to predict software systems' success or failure.
RQ3. Do practitioners consider misinterpretation of requirements, and communication errors are the main causes behind software failure?	Q12, Q15, Q16, Q17, and Q18 are direct questions that highlights the software practitioners' opinions about the role of requirements' misinterpretation, and communication errors that contribute to decreasing a software system's success.

(Continued)

Table 2: Continued

Research questions and goals	Motivation
RQ4. What are the means used by practitioners to predict the success or failure of software systems?	Q2, Q3, Q5, Q13, and Q14 are direct questions about decision making regarding software success or failure rate forecasting. Self-descriptiveness, better information judgment, right decision making and their effects, and avoiding reworks are the primary tools considered in decision making.

3.2 Participants

Over 104 industry practitioners were approached in different software companies and institutes using an online data collection tool (inqwise.com). This empirical study was based on the software development companies and academic institutes in Pakistan. According to the 'global connectivity index' (GCI), Pakistan ranked in 76 positions in 2019. These companies are listed in the following. Of the 104 people, 60 answered and completed the questionnaire and returned within two weeks. Therefore, the final analysis was performed with a 58% response rate from participants. However, we received 08 questions mostly filled by study's participants that were included for analysis. Data were checked for correctness, completeness, and distributions. Data information was converted into counts and, percentages via regular descriptive statistics. Next, we compared the participants from these following organizations, who stated that cognitive bias impacts the success of software.

- Makabu Islamabad
- Pakistan Revenue Automation Limited
- PTCL (Pakistan Telecommunication Limited)
- Quality Web Programmer
- FJWU
- Discretelogix
- WADIC
- NUST

3.3 Data Collection Procedure

A questionnaire is applied to collect information from professionals of the software industry population. An online survey is a data collection method that accomplishes a research study's purposes using the questionnaire or interview technique. A researcher uses the online empirical data collection technique to collect data from participants in their opinions, attitudes, effects, and relationships. Primarily purpose of the questionnaire is the answering of research problems in a quantitative study by using the 5-Point Likert Scale. Data is collected in the spread-sheet tool of Microsoft Excel that is exported from the online survey tool [inqwise.com]. Data collected in the excel sheet is further refined to get useful information.

3.4 Data Analysis

We sent online questionnaire to a total of one hundred and four participants. Out of these:

- 60 responses were completed
- Four responses were sent to us blank, indicating that participants have seen the questionnaire page and did not answer any question.
- Twenty-two respondents partially completed questionnaires. The mean time for the completion of the questionnaire was 20 min.

An examination of the partially answered questionnaire revealed that they contributed very little. Several respondents did not mention their designation and experience while answering the survey questionnaire. Toa analyze the collected information, we performed t-test to compare the statistically relationship between variables. Data collected against the RQs was analyzed using the statistical t-test by considering *p*-values of each RQ in this empirical study.

4 Empirical Study's Results and Findings

This section presents empirical study's results and their findings.

4.1 The Respondents (RQ1-RQ3)

As earlier stated that a total of sixty-eight respondents completed the survey questionnaire: 56 from Pakistan, two from each country such as Bahrain, Netherland, United States, United Kingdom, and one from each of countries such as Saudi Arabia, Malaysia, Bahrain, Denmark, and one respondent did not mention the location.

Fig. 2 shows that 29% of the respondents stated their experience two or less than two years. 32% of respondents had experienced between two years and four years, 22% of respondents were experienced more than four years and less than eight years, and 17% of the respondents were experienced more than eight years.

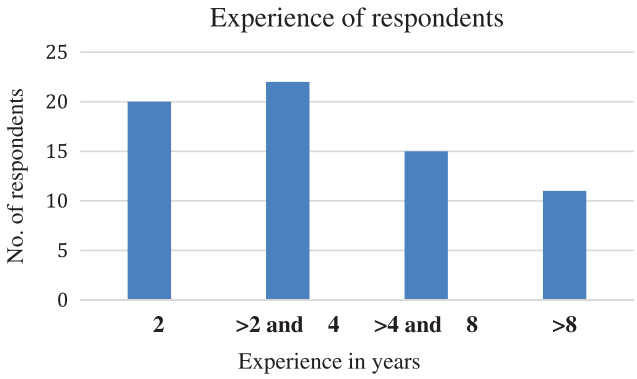

Figure 2: Experience in years

Fig. 3 shows the role of respondents they play in the software development industry. 27 out of 68 respondents have programming responsibilities in their workplaces. Nine respondents were playing their roles as system analysts. Ten respondents were involved in requirement engineering processes. Seven were project managers, and their responses indicated that they had a technical lead, rather than the administrative role. A large number of respondents (14 respondents) were from different fields in the software industry. Only three respondents did not mention their designation.

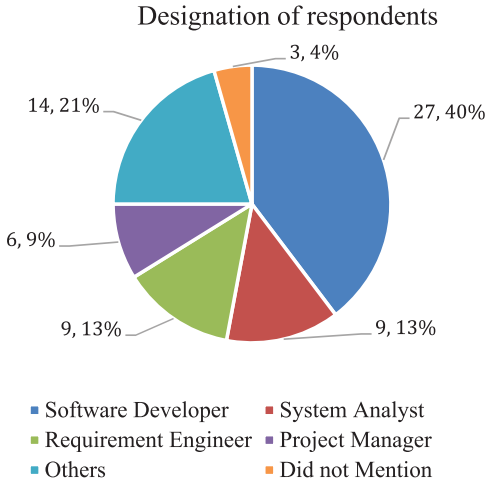

Figure 3: Respondents' designations

Non-probabilistic sampling is the main technique used in this study. The findings, although it cannot be generalized to the broader population, respondents seem to represent the diverse groups of experienced people from the software industry. Therefore, the common findings indicate the respondents are from the focused population and show the best approach to forecast the success or failure rate of software projects.

4.2 Human Factors (RQ1)

We asked study's participants whether precision in software success or failure rate forecasting was associated with the accurate functioning of a software system (Q1). 16 or (27%) respondents rate Q1 as strongly disagree and disagree, 25 or (42%) respondents rate as agree and strongly disagree, and a large proportion (19% or 32%) of respondents remain neutral while answering Q1.

The empirical study's question (Q4) is asked from participants to know whether software success forecasting from human factors is reliable. 23 or (34%) of the responses rate strongly disagree and disagree, 26 or (39%) rate agree and strongly agree, and 18 or (27%) responses from respondents neither disagree nor agree with the statement.

The empirical study's question (Q6) is purely based on the cognitive bias as a human concern. Cognitive bias is usually a deviation pattern in decision-making or judgment, in particular, situations [27,28]**.** Deviation pattern is the standard of comparisons that may be a set of verifiable independent facts or people's judgment outside of specific circumstances. Human factors in psychological and physical forms affect software project development [10]. To know whether cognitive bias has an impact on human judgment in forecasting software success or failure. 16 or (26%) of responses rate strongly disagree and disagree, 26 or (42%) responses rate agree and strongly agree, and 20 or (32%) responses are neutral in answering Q6.

As shown in Fig. 4, 19, or (29%) of the responses rate strongly disagree and disagree, 23 or (35%) of responses rate agree and strongly agree, and 24 or (36%) of the responses remain neutral while answering Q7. To answer Q8, 15 or (25%) responses rate strongly disagree and disagree, 27 or (45%) rate agree and strongly agree, and 16 or (30%) responses remain neutral.

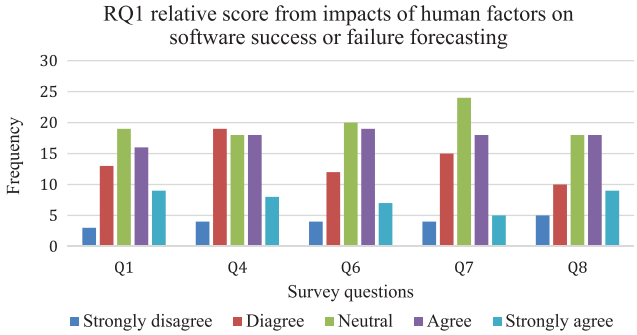

Figure 4: RQ1's relative score from human factors and their impacts

Further correlation between themes stated in (Q1, Q4, and Q6-Q8) and their relevancy as human factors to software success estimation can be statistically measured. Thus, a paired sample t-test is a statistical method to find a correlation between two variables.

Tab. 3 summarizes the t-value and *p*-value of statistically significant at an error level of 5 percent from construct variables (Q1, Q4, Q6, Q7, and Q8) and human factors. Studies [29,30] find that t-value ≥ 1.96 is known as significant for $p \leq 0.05$. This empirical study considered the 5 percent resulting errors.

Table 3: Relationship based on t-value and *p*-value

Relationship	t-value	Statistically significant	*p*-value
Q1-Human factor	4.297	Yes	0.013
Q4-Human factor	4.334	Yes	0.012
Q6-Human factor	3.910	Yes	0.017
Q7-Human factor	3.438	Yes	0.026
Q8-Human factor	4.636	Yes	0.014

Key Finding 1 (KF1): The majority of respondents from the software industry have considered that human factors, including cognitive bias, have negative impacts on software success or failure forecasting in software engineering.

4.3 Recognition of the Accurate Phase (RQ2)

The summary of responses to Q9-Q11 on the significance and recognition of the phase for software success or failure forecasting shows a mixed kind of responses. A theme that emerges from the responses to Q9 is difficult in answering the respective empirical study's question. The selection of the requirement gathering phase, an *accurate phase* for software success rate forecasting, is not supported by a majority of respondents. If we look at Fig. 4, 23 or 37% of respondents rate it strongly disagree or disagree, 25 or (40%) of respondents rate it agree and strongly disagree, and 15 or (23%) respondents remain neutral while answering question Q9 of this empirical study.

Fig. 5 shows us further results of Q10 and Q11 from study's participants. Survey question Q10 seeks respondents' answers to know about the software success or failure rate forecasting from software specifications. 27 or (43%) respondents do not show their positive response as they rate it strongly

disagree or disagree, 21 or (33%) respondents show positive responses by rating it agree and strongly agree. A large portion of respondents (18% or 29%) remain neutral while answering Q10. For question Q11, we have positive responses from 31 or (50%) of respondents. 17 or (27%) respondents rate it strongly disagree and agree, and 14 or (23%) respondents remain neutral while answering Q11.

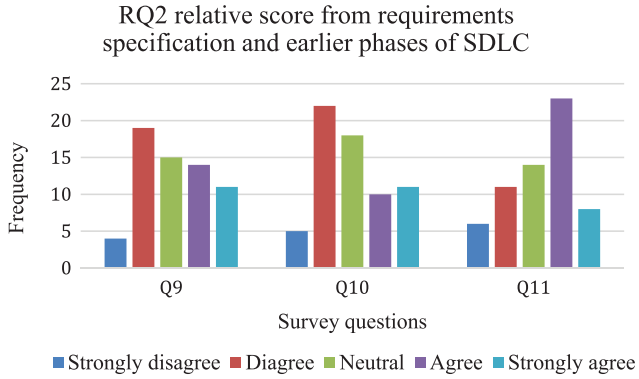

Figure 5: RQ2's relative score from the recognition of the accurate phase

Key Finding 2 (KF2): Earlier phases of SDLC were identified as important phases of software systems for their success or failure prediction. The SDLC's requirement engineering phase is not only predicted as an accurate phase to detect software success or failure but also saves a lot of cost on software re-development. Therefore, requirement specifications are considered the means of determining the software success rate.

4.4 Requirements' Misinterpretation, and Communication Errors (RQ3)

To know about the misinterpretation of requirements at the earliest phase of the SDLC, Q12 was asked from participants of this study. Moreover, survey questions Q15-Q18 were asked to know whether communication errors were among the software failure causes. Survey questions (Q15-Q18 and Q12) are directly associated with RQ3 in this study. The RQ3 is aimed to seek the direct answer behind the failure of software systems.

A recent work [31] reveals that conflicts arise from the misinterpreted elicited requirements. Conflicts are challenging for every type of software system. Hence misinterpretation of requirements can occur intentionally or unintentionally while implementing them into business processes.

Fig. 6 shows positive support from respondents about the misinterpretation of requirements as the primary cause behind software failure. 32 or (54%) respondents rate it agree and strongly agree; only 16 respondents (27%) rate it strongly disagree and disagree, while 11 or (19%) of the respondents remain neutral in answering the Q12.

To know whether the communication is one of the causes behind software failure, Q15 was answered by 63 respondents. Twenty-seven of the total respondents or (43%) rate it agree and strongly disagree, 20 or (32%) of the total respondents rate it disagree and strongly disagree, and 16 or (25%) of the total respondents remain neutral.

Two questions (Q16 and Q17) of this empirical study were focused on poor communications and their impacts on software estimation and quality requirements. Let's look at the responses of participants in answering Q16. We find that majority of respondents 39 or (62%) rate it agree and

strongly agree, 23 or (24%) respondents rate it strongly disagree and disagree, and 9 or (14%) of the respondents remain neutral while answering Q16.

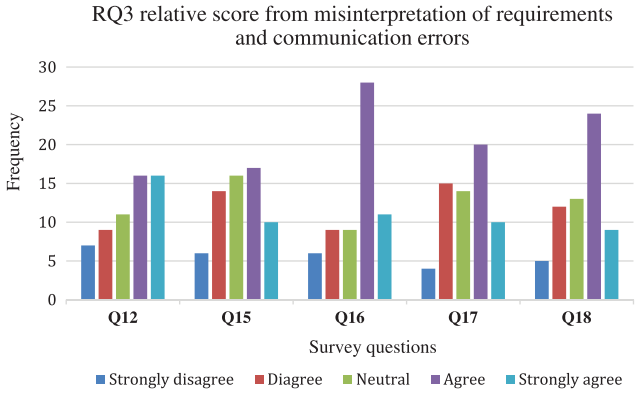

Figure 6: RQ3's relative score from requirements' interpretation and miscommunication

As shown in Fig. 6, 30 respondents answered agree and strongly agree in 48% of cases, 19 respondents answered strongly disagree and disagree in 30% cases, and 14 respondents remained neutral in 22% cases in answering Q17. Our empirical study results justify that software practitioners are more positive in revealing that poor communication between software clients and requirement engineers has impact on software estimation. Survey question Q17 focuses on quality requirements, which emerge from strong communication between clients and software practitioners.

Survey questions Q18 aimed to seek answers from participants regarding the relationship between poor communications and delayed software specifications. 33 or (52%) respondents remained positive while rating it agree and strongly agree, 17 or (27%) remained negative as they rated it strongly disagree and disagree, and 13 or (21%) respondents remained neutral while answering the Q18.

Key Finding 3 (KF3): Requirements' misinterpretation was identified as the important cause behind the failure of software systems by the majority of respondents (54%). Poor communication between clients and requirements engineers was also identified as the major impacting element on software estimation or prediction because the majority of respondents (62%) remained positive. Respondents warned the software practitioners against poor communication; most of them (52%) identified poor communication that delayed the software specifications.

4.5 Decision Making (RQ4)

This paper's RQ4 was focused on decision making for better software success or failure forecasting. There were five survey questions (Q2, Q3, Q5, Q13, and Q14) to answer the RQ4. We have presented the score of each survey question in the following Fig. 7.

Fig. 7 shows us that most responses (45%) agree and strongly agree, followed by 35% responses that do not rate Q2 either agree or disagree. In contrast to the earlier mentioned responses, 20% of the responses rate strongly disagree and disagree while answering Q2. Survey participants were asked to reveal their opinion about the importance of better judgment for accurate software success forecasting or failure. In reply, 18 or (30%) of the responses rate strongly disagree and disagree, 30 or (49%) of the responses rate agree and strongly agree, followed by 13 or (21%) of responses that remain neutral while answering Q3.

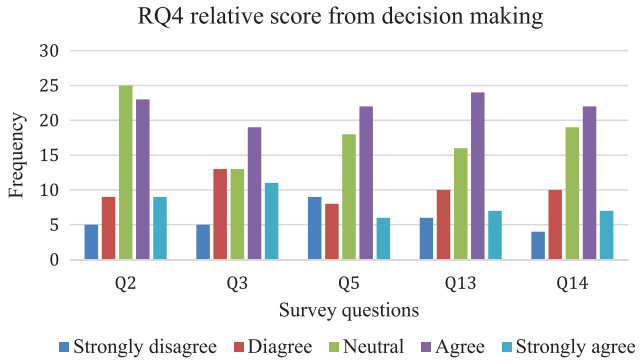

Figure 7: RQ4's relative score from decision making and related constructs

Survey participants were asked to state their opinion by rating Q5, whether software success or failure estimation leads to the right direction for expanding software requirements. Requirements specification has a vital role in the final quality product. Quality requirements have quality factors, including maintainability, portability, and expandability [32]. Therefore, the right choice to prioritize quality requirements over other requirements can be a better decision for quality requirements [33]. We were motivated by these studies and thus included Q5 in the survey questionnaire. 17 or (27%) respondents did agree with our assumption. In contrast, 28 or (44%) respondents were agreed with our statement, followed by 16 or (29%) respondents who were neutral while answering survey question Q5.

Information judgment is reliable, relevant, and trustworthy in making accurate decision-making [34]. Our survey study identifies a challenge for researchers to determine the exact information type that helps make better software success or failure prediction. We find that 16 or (25%) respondents did not agree with our statement from survey results, as their responses rated strongly disagree or disagree. However, 31 or (49%) of the respondents agree with us by rating their responses agree and strongly agree. 16 or (25%) respondents did not agree with the above statements, and they remain neutral once they answered Q13 from the survey questionnaire. The survey question (Q14) is focused on the implementation of the software success or failure estimation and its outcomes regarding rework avoidance on software systems. Rework performed on the software in the lateral phase is more expensive than the design work performed in the earlier phases of SDLC [35,36]. We find that 14 or (23%) respondents were not agreed with our statement to know about software practitioners' rating. In contrast, 29 or (47%) agreed with our statement, followed by 19 or (31%) respondents who were neutral in answering the Q14 from the questionnaire survey.

Key Finding 4 (KF4): Overall, respondents state their positive rating in answering RQ4. The majority of respondents call better information judgment as a means of accurate software success or failure forecasting.

5 Limitations

This research study has a limitation regarding data from a survey that cannot be generalized to global software practitioners in the industry. This research represents the software practitioners' community from the city of Islamabad, Pakistan. The majority of respondents cannot be considered a novice as they have four or fewer years of experience working in software development companies. We found that many respondents did not rate the survey questionnaire, and they are moderately knowledgeable about software success or failure estimation.

Several constructs that surround the four main research questions may create confusion among the respondents while answering the survey questionnaire. Research questions (RQ1-RQ4) were defined with many other related constructs, and respondents may have faced difficulty filling the survey questionnaire.

In this survey study, we have recognized numerous threats to validity. Regarding construct validity, the biggest issue is the choice of a research topic explicitly covered by the survey. It may be that other related software prediction topics also have been explored. At the time of the questionnaire design, little indication of the important research topics was omitted.

6 Conclusion

Summarizing the results of this empirical study, it has been seen that human factors impact while predicting software success or failure. Cognitive bias has been identified as one of the major human factors that negatively impacts forecasting software success. Our empirical study also finds that earlier phases of SDLC are better in predicting the software success or failure and help prevent the high expenses on rework on the later software development phases. Our empirical study's results show that misinterpretation and miscommunication of requirements are among the causes behind software failure. Finally, this empirical study's results show the importance of decision making from reliable, relevant, and trustworthy sources of information for software success or failure prediction.

Funding Statement: This research was supported by the BK21 FOUR (Fostering Outstanding Universities for Research) funded by the Ministry of Education and National Research Foundation of Korea.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] A. Bazhenov and V. Itsykson, "Forecasting software development project characteristics using meta-modeling," in *Proc. of the 9th Central & Eastern European Software Engineering Conf.*, New York, NY, USA, pp. 1–8, 2013.

[2] L. Chen, B. Fang, Z. Shang and Y. Tang, "Tackling class overlap and imbalance problems in software defect prediction," *Software Quality Journal*, vol. 26, no. 1, pp. 97–125, 2018.

[3] D. Bowes, T. Hall and J. Petrić, "Software defect prediction: Do different classifiers find the same defects?," *Software Quality Journal*, vol. 26, no. 2, pp. 525–552, 2018.

[4] Y. Hu, B. Fing, X. Mo, X. Zhang, E. W. T. Ngai *et al.,* "Cost-sensitive and ensemble-based prediction model for outsourced software project risk prediction," *Decision Support Systems*, vol. 72, pp. 11–23, 2015.

[5] P. Pospieszny, "Software estimation: Towards prescriptive analytics," in *Proc. of the 27th Int. Workshop on Software Measurement and 12th Int. Conf. on Software Process and Product Measurement*, New York, NY, USA, pp. 221–226, 2017.

[6] 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, no. 2, pp. 181–193, 2007.

[7] E. V. Kelle, J. Visser, A. Plaat and P. V. D. Wijst, "An empirical study into social success factors for agile software development," in *Proc. 2015 IEEE/ACM 8th Int. Workshop on Cooperative and Human Aspects of Software Engineering*, Florence, Italy, pp. 77–80, 2015.

[8] N. Ali, S. Hwang and J.-E. Hong, "Your opinions let us know: Mining social network sites to evolve software product lines," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 13, no. 8, pp. 4191–4211, 2019.
[9] Y. Shi, M. Li, S. Arndt and C. Smidts, "Metric-based software reliability prediction approach and its application," *Empirical Software Engineering*, vol. 22, no. 4, pp. 1579–1633, 2017.
[10] N. Chotisarn and N. Prompoon, "Forecasting software damage rate from cognitive bias in software requirements gathering and specification process," in *Proc. 2013 IEEE Third Int. Conf. on Information Science and Technology (ICIST)*, Yangzhou, China, pp. 951–956, 2013.
[11] 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, no. 1, pp. 144–160, 2013.
[12] R. Noor and M. F. Khan, "Defect management in agile software development," *International Journal of Modern Education and Computer Science*, vol. 6, no. 3, pp. 55–60, 2014.
[13] P. C. Véras, E. Villani, A. M. Ambrosio, N. Silva, M. Vieira *et al.,* "Errors on space software requirements: A field study and application scenarios," in *Proc. 2010 IEEE 21st Int. Symp. on Software Reliability Engineering*, San Jose, CA, USA, pp. 61–70, 2010.
[14] M. Kang, O. Choi, J.-H. Shin and J. Baik, "Improvement of software reliability estimation accuracy with consideration of failure removal effort," *International Journal of Networked Distributed Computing*, vol. 1, no. 1, pp. 25–36, 2013.
[15] H. Hewamalage, C. Bergmeir and K. Bandara, "Recurrent neural networks for time series forecasting: Current status and future directions," *International Journal of Forecasting*, vol. 37, no. 1, pp. 388–427, 2021.
[16] S. Sinha, N. K. Goyal and R. Mall, "Early prediction of reliability and availability of combined hardware-software systems based on functional failures," *Journal of Systems Architecture*, vol. 92, pp. 23–38, 2019.
[17] I. Saidani, A. Ouni, M. Chouchen and M. W. Mkaouer, "Predicting continuous integration build failures using evolutionary search," *Information and Software Technology*, vol. 128, pp. 1–16, 2020.
[18] J. Moon, S. Kim, J. Song and K. Kim, "Study on machine learning techniques for malware classification and detection," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 15, no. 12, pp. 4308–4325, 2021.
[19] L. Zhao and Y. Zhang, "Generative adversarial networks for single image with high quality image," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 15, no. 12, pp. 4326–4344, 2021.
[20] S. Li, Q. Zhou and W. Wei, "Malware detection with directed cyclic graph and weight merging," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 15, no. 9, pp. 3258–3273, 2021.
[21] Y. Alotaibi, "Automated business process modelling for analyzing sustainable system requirements engineering," in *6th Int. Conf. on Information Management (ICIM)*, London, UK, pp. 157–161, 2020.
[22] T. Ambreen, N. Ikram, M. Usman and M. Niazi, "Empirical research in requirements engineering: Trends and opportunities," *Requirements Engineering*, vol. 23, no. 1, pp. 63–95, 2018.
[23] H. H. Khan, M. N. Malik, Y. Alotaibi, A. Alsufyani and S. Alghamdi, "Crowdsourced requirements engineering challenges and solutions: A software industry perspective," *Computer Systems Science and Engineering*, vol. 39, no. 2, pp. 221–236, 2021.
[24] Y. Alotaibi, "A new secured e-government efficiency model for sustainable services provision," *Journal of Information Security and Cybercrimes Research*, vol. 3, no. 1, pp. 75–96, 2020.
[25] A. Arshad, S. Riaz, L. Jiao and A. Murthy, "The empirical study of semi-supervised deep fuzzy c-mean clustering for software fault prediction," *IEEE Access*, vol. 6, pp. 47047–47061, 2018.
[26] Y. Alotaibi and A. F. Subahi, "New goal-oriented requirements extraction framework for e-health services: A case study of diagnostic testing during the COVID-19 outbreak," *Business Process Management Journal*, vol. 28, no. 1, pp. 273–292, 2021.
[27] F. Shull, J. Singer and D. I. Sjoberg, *Guide to Advanced Empirical Software Engineering*, London, UK: Springer, 2007. [Online]. Available: https://link.springer.com/book/10.1007/978-1-84800-044-5#about.
[28] P. E. Strandberg, E. P. Enoiu, W. Afzal, D. Sundmark and R. Feldt, "Information flow in software testing–an interview study with embedded software engineering practitioners," *IEEE Access*, vol. 7, pp. 46434–46453, 2019.

[29] R. Mohanani, I. Salman, B. Turhan, P. Rodríguez and P. Ralph, "Cognitive biases in software engineering: A systematic mapping study," *IEEE Transactions on Software Engineering*, vol. 46, no. 12, pp. 1318–1339, 2018.
[30] J. F. Hair, C. M. Ringle and M. Sarstedt, "PLS-SEM: Indeed a silver bullet," *Journal of Marketing Theory and Practice*, vol. 19, no. 2, pp. 139–152, 2011.
[31] K. K.-K. Wong, "Partial least squares structural equation modelling (PLS-SEM) techniques using SmartPLS," *Marketing Bulletin*, vol. 24, no. 1, pp. 1–32, 2013.
[32] Q. Ramadan, D. Struber, M. Salnitri, J. Jurjens, V. Riediger *et al.,* "A Semi-automated BPMN-based framework for detecting conflicts between security, data-minimization, and fairness requirements," *Software and Systems Modelling*, vol. 19, no. 5, pp. 1191–1227, 2020.
[33] C. Wang, M. Xu, W. Wan, J. Wiang, L. Meng *et al.,* "Robust image watermarking via perceptual structural regularity-based JND model," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 13, no. 2, pp. 1080–1099, 2019.
[34] C. Rosen, "Software systems quality assurance and evaluation," in *Guide to Software Systems Development*, Switzerland: Springer, pp. 101–123, 2020.
[35] M. Park and P. Oh, "Judgment making with conflicting information in social media: The second-order judgment problems," in *Proc. Int. Conf. on Social Computing and Social Media*, Toronto, Canada, pp. 141–150, 2016.
[36] B. M. Kennedy, D. K. Sobek and M. N. Kennedy, "Reducing rework by applying set-based practices early in the systems engineering process," *Systems Engineering*, vol. 17, no. 3, pp. 278–296, 2014.