JOURNAL OF SOFTWARE ENGINEERING & INTELLIGENT SYSTEMS
ISSN 2518-8739
31st December 2016, Volume 1, Issue 2, JSEIS, CADMEI Copyright © 2016
www.jseis.org

The RATE framework development from cognitive bias during requirement engineering processes

[1]Muhammad Hassnain, [2]Imran Ghani
[1]Army Public College of Management & Sciences, Rawalpindi, Pakistan
[2]Monash University, Malaysia
Email: [1]hassnain.pk77@gmail.com, [2]imran.ghani@monash.edu

ABSTRACT

Development of a framework for system's reliability, accuracy, tolerance and estimation in the context of human factors is a substantial work. This research is aimed to see the impact of human factors on the requirement engineering processes. Proposed reliability, accuracy, tolerance and estimation (RATE) framework is composed of four components accompanied by the relevant sub-components. We aim to present the nominal scoring of all sixteen sub-components that involve during requirement engineering processes. Method execution provides the significant values of all sub-components and their retention in the framework. Results show that stakeholders of software industry have concerns from cognitive bias throughout the requirement engineering processes. Four components and relevant sub-components find their retention in framework as study participants also respond to keep these components and sub-components in the refined RATE framework.

Keywords: RATE framework; cognitive bias; software damaging; requirement engineering processes; reliability; estimation;

1. INTRODUCTION

Human design errors occur during software designing, software coding and human misinterpretation of requirements. Total numbers of faults made through software development life cycle are unknown to end users and they require estimation of these errors or faults. An error needs observation for detection before it is fixed. Software failures result into system failure, which are software errors. Corrected errors are errors, which are fixed but never integrated into current software version. Verified errors are corrected errors that are also verified for current released version of software [1]. Project management and software development are the knowledge intensive as human factors are involved in the each link. Fault forecasting is the more significant task that requires practical determination in the software engineering discipline. If a method of software damage rate forecasting is developed and applied, it can serve as a tool for a reliable software creation. To measure the faults created from human factors during requirement engineering is a popular research area that requires framing the cognitive bias, activities, errors or attributes in the non-functional requirements of a system. Determination of impact values of individual cognitive bias, errors or attributes and activities during the requirement engineering process is the concerning area of this research. This paper will examine the development of RATE framework; correlate the subcomponents of reliability, accuracy, tolerance and estimation components. The RATE framework will be evaluated by the results and their interpretation from participants' responds in the software development industry.

Software success or failure depends upon the decision making in such circumstances that may result into distorted perception and incorrect judgment. Correct decision making if found lacking during requirement engineering processes may result into software failures. Early software damage rate forecasting from cognitive bias, errors and activities requires a special attention of researchers. Software designers cannot design sound software projects if cognitive bias are left to be considered.

2. RELATED WORKS

This section aims to review the related literature on research topic and explores the cognitive bias during the software requirement engineering processes. It is also aimed to review the previous research studies on the software estimation frameworks and models that incorporate the human factors.

2.1 Cognitive bias types

Mental or cognitions behaviors, which prejudice the quality of decision for a number of judgments or decision making were called as cognitive biases. In a situation where tendency of selecting the pre-selected options over the

unselected are superior one option has been known as a *default bias*. Wilkenson and Klaesin in [2] defined the cognitive bias as systematic errors found in the human decision-making. Various cognitive biases were identified, but most of the research studies considered only one or two human factors. From previous research studies, some of the cognitive biases were found as the overlapping. Unrealistic optimist bias was resembled closely to valence effect in meaning. Because of the interconnected and similar biases, complexes of biases were worked out. Biasplexes had positive as well negative consequences on the project development [3]. Software defects were also caused by the pattern deviation found in human thought from laws of mathematics and logic. This assertion has been substantiated by the little empirical evidence to date [4].

2.2 Cognitive system and human decision making

In a research work [5] Zhang et. al, examined two cognitive bias factors that appeared in the loss aversion, conservatism and decision making in the real-world scenarios. Conservatism revealed that beliefs were adjusted insufficiently on appearance of new information. Judgment accuracy was affected by this type of bias. It also overestimated the low probabilities and underestimated the high probabilities. This bias resulted into an inaccurate decision making in real world scenarios. Impacts of loss aversion were shown that influenced the decision making. People tendency towards the disproportional preferring for loss avoidance was known as loss aversion that aimed for acquiring the gains. It was found that two cognitive biases co-existed in most situations and impacted the decision-making of people. In the same work, two estimation techniques, expected return (ER) and bootstrap were explored**.** Expected return technique helped the participants of study to show rationality in decision making while bootstrap made participants to become more consistent. Sometimes, such types of selection coincided the situations in a real world. Decision making became costly due to such cognitive bias. This work was one of the initial steps towards the development of cognitive systems, which could improve the human decision making.

2.3 Fuzzy set theory

Gonzalez-Carrasco et al., in [6] found that theory of fuzzy sets only provided the information about uncertainty from human knowledge. Real time systems have embedded the use of Fuzzy set theory for different areas like decision making domains. Since, engineers and researchers preferred the use of fuzzy set theory. However, inherent uncertainties have produced the concerns of applications from using the fuzzy logic. Effort estimation was called the base for the accurate estimation of software systems. Therefore, project management in sense of minimizing the risk and better resource allocation is best facilitated by the effort estimation.

Liu et al., noticed that root cause of modules' failure was the human error. In order to overcome this issue, diversity in human factors played the effective role. Benefits obtained from diversity in human factors were difficult to quantify them. They developed an approach of human reliability analysis. This approach was based on a fuzzy logic model that mapped the common causes of a failure to a component degradation value [7]. Effort estimation for software development has remained very low that leads towards the poor management of project failures and plans. This bias is caused because information affects the software developers and information revealed is not relevant to the actual effort estimation. In a survey work [8] three sets of tests included participation as 374 developers from various outsourcing companies. They determined the connection between developers' dimensions and estimation along several other factors of dimensions like thinking style, self-construal, skills, education, nationality and experience. It was found that most of studies' dimensions were connected with estimation bias. At the higher interdependence, the estimation bias was found to be increased.

2.4 Naive bayes classifier

Software estimation and prediction approaches using the quantitative based models were main contributions to control project risk at the early phase of software development life cycle. Uncertainty and instability concerned the software projects, but early estimation and prediction of limited data could be updated by the increment of new acquired data during the execution of data. Naive Bayes Classifier was tested for the effectiveness over the data collected from 104 projects. Application of Naive Bayes Classifier was compared to the Poisson regression model at all phases of the project development life cycle. Results of the experiments showed that Naive Bayes Classifier was more accurate than Poisson regression model to determine the area under curve (AUC). However, this model was limited when compared to other approaches with the different data sets [9].

2.5 Mental model

Mental model was designed for keeping the concept of success. Success could be only achieved through contribution towards the achievement of success. Economic gain was the main objective of success that could not be controlled by project leaders and developers. Software developers could not reduce the impacts of political influence, and inappropriate deadlines. A concept of requirement compliance was introduced that measured the size of interaction among the defined and implemented requirements. Success was measured from the implemented requirements. Contrary to implemented requirements, all other unsolicited features and uncompleted requirements reduced the success of software projects [10].

2.6 Software requirement engineering processes

In [11] Mohanani et al., called that requirement engineering was referred to the collection of activities with the academic field where these activities were studied. Requirement engineering has broad definition that includes the activities like understanding, documenting, specifications, communication and modification of problems. Other aspects of requirement engineering included the users, goals, non-functional requirements and agents. In the same study researchers also investigated that designers' creativity was affected by requirements. Minor changes in requirement engineering processes possessed the potential power due to sensitivity of designers. Designers' sensitivity was related with framing effects of cognitive biases. Emotions were reported as key concerns for people behavior. Software engineering processes were called as the intensive and capital activity of human; hence emotion management was obviously a software profession. It was found that emotions were key factors, which were taken into account for keeping the stability in software requirement [12].

Marnewick et al., in [13] investigated the factors involved in delivering the poor quality of requirements. Human factors were found to be affecting the quality of requirements. An alternative view of communication skill was suggested to compensate the poor quality of requirements by trust relationship between customers and requirement engineers. This alternative view enabled the requirement engineers to elicit the high quality requirements from users. Communication as a human factor could not be resolved through the requirement solution. It was gained by talking with the experienced persons, because different people used a variety of communication tools. If a requirement engineer lacked the ability to communicate with customers that created the issue because requirement engineer was responsible to initiate the communication.

3. PROPOSED MODEL

Idea behind the development of Reliability, Accuracy, Tolerance and Estimation (RATE) framework is to include the quality attributes of a software measurement in terms of cognitive bias impacts. RATE framework implies the Reliability, Accuracy, Tolerance and Estimation components. Proposed model is designed after a thorough literature review on cognitive bias role in software requirement engineering. Designed framework is aimed to forecast the damage rate produced from cognitive bias. The RATE framework as self-explanatory is intended to investigate the software reliability, accuracy, tolerance, and forecasting the software damage rate. Description of its components and sub-components is given in Table 1. There is a single sequence among all components of RATE framework that does not mean software is struck if any cognitive bias, requirement errors, quality attributes and activities performed are not measured.

In Figure 1, four components of framework with their sub-components are shown. Software reliability is acquired through its all-time availability with self-descriptiveness, expandability and free of damaging. Software accuracy is acquired if communication error during requirement engineering process of requirement elicitation is avoided or handled properly. However, communication error handling alone cannot guarantee that a system is accurate. Elicited requirements need to be interpreted accurately to make the system's working more accurate. Requirement interpretation can become impacted from anchoring and affects the software accuracy. Communication error, misinterpretation of requirement and anchoring achieved signify that how software is precise in its performance. Reliability and accuracy components achievement also requires the software tolerance as a separate requirement of a system as shown in RATE framework diagram. This component is specified towards the cognitive concerns, human bias and software perception. These factors impact the system tolerance that requires the neutralization of the impacts produced from such factors in the serious circumstances. In the final component of RATE framework, one requires to estimates these sub-components of all three components. Forecasting is achieved if socio-technical intervention and judgment are used fairly in order to make a decision. This framework will be

introduced in a number of software development organizations with the aim to structure the requirement engineering processes. This framework will work as a reference to train the requirement engineers, software analysts, developers and managers. The goal of measuring the human bias impacts during requirements engineering processes remains the main feature of proposed RATE framework. As the sequence shown in the Figure 1 our framework involves the requirement engineering processes. It encompasses the non-functional requirements; each NFR is related to four sub-components. The sub-components generalize their importance when requirement engineering processes are continuing. The degree of human bias, errors produced and activities involved in requirement elicitation, analysis, validation and documentation are covered. This framework uses the means to measure the impact value of cognitive bias 'availability' under the reliability component. Cognitive availability directly impacts the requirement elicitation processes. To test this proposition the, framework measures how cognitive availability influences the core non-functional features like self-descriptiveness and expandability. If cognitive availability is measured with the value more than 0.60, it has effects on the self-descriptiveness and expandability that results into damaging of software reliability. Anchoring bias in Accuracy component leads towards the appropriate solution but impacts from human errors in communications and misinterpretations of requirements during requirement elicitation, analysis and validation processes. Framework shows that if anchoring bias is adversely impacted from communication error and misinterpretation of requirements, precision in results of software is reduced. Cognitive concerns and human bias directly impact the software fault-tolerance capability.

Framework measures the human bias, cognitive concerns and software size perception; if value exceeds than 0.60 the fault-tolerance NFR of a system is impacted. Estimation is a vital component of framework that forecasts the values of preceding components and sub-components. Cognitive-bias affects the judgment in decision making. Because, requirement engineering processes involve the correct judgment and decision making of clients as well as requirement engineers, socio-technical interventions overcome the problems produced from the cognitive bias.

Table. 1 Description of RATE framework components and sub-components

Components	Description with sub-components
Reliability	Reliability of system is fundamental non-functional requirement. Software reliability is derived from its sub-components of availability, self-descriptiveness, expandability and damaging. Software is reliable if it is available 24 hours/day. When software availability is ensured, it must show self-descriptiveness, and expandability features. Combined software availability, self-descriptiveness and expandability make the system free of any damaging about system reliability.
Accuracy	Accuracy component is linked to preceding component of reliability, because reliable software is ensured to show the maximum accuracy. However, software has accuracy concerns due to communication errors between requirement engineers and clients, requirement misinterpretation and anchoring bias. In order to remove, say that 10% accuracy error, communication errors, requirement misinterpretation and anchoring bias concerns are handled to attain the 100% software accuracy.
Tolerance	Software tolerance is linked to preceding framework component called accuracy. Software is tolerant if accuracy and reliability aspects are attained. However, system tolerance is impacted from cognitive concerns, human bias and software size perception. To overcome these impacting factors, input neutralization is applied in extreme situations to increase the software tolerance.
Forecasting	Software reliability, accuracy and tolerance achieved through earlier three components of RATE framework are required to be quantified and estimated to forecast the damage rate if it exists there. Estimation component is dependent upon forecasting activity, socio-technical interventions, and judgment. Applying these factors properly, a best decision-making can result into RATE-ed requirements.

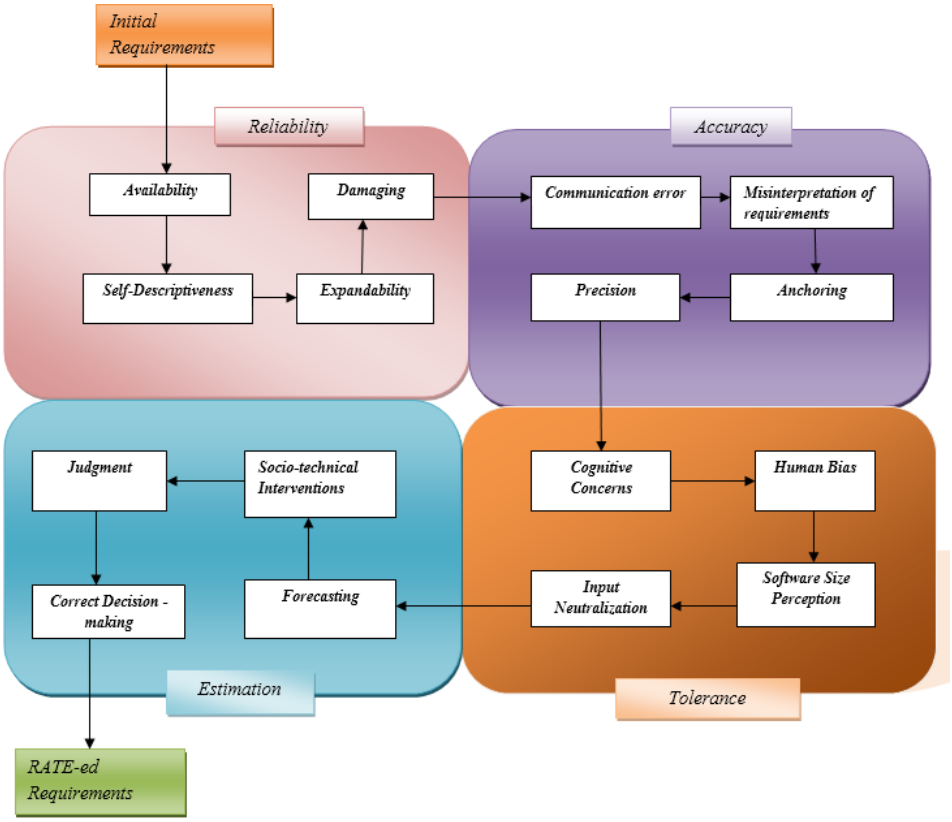

Figure. 1 RATE framework

4. RESEARCH METHODOLOGY

Primary aims behind the current research is to develop and understand the RATE framework from existing literature; use the framework to measure the cognitive bias, activities, and errors made during requirement engineering processes. A quantitative research method has been selected that concerns the responds of professionals from software industry.

4.1 Hypothesis development

Hypothesis is based on the nominal scoring analysis as presented in the following:

H0: Cognitive bias, errors and activities measured from non-functional requirements have no effects on the software development lifecycle.

H1: Cognitive bias, errors and activities measured from non-functional requirement have effects on the software development life cycle.

If H1 is accepted then attribute or errors estimation, activities measurement during requirement engineering processes informs about their impacts on the software quality.

4.2 Data collection

Data has been collected from several experienced software requirement engineers, system analysts and project managers from the following software development organizations operating in Pakistan.

1. MAKABU Islamabad
2. Quality Web Programmer
3. Discretelogix
4. WADIC

5. Pakistan Revenue Automation Limited

All of the research study participants were sent invitation in the context of research objectives. When they accepted the invitation, the link of questionnaire was emailed to all of the participants.

4.3 Survey technique

Online survey has been designed to collect information from target participants through a questionnaire. A comprehensive questionnaire includes the propositions about cognitive bias impacts, errors or attributes and related activities during requirement engineering processes as given in the Appendix B. Participants' opinions about causes and impacts of software quality attribute are obtained by using the 5-Point Likert Scale.

Table. 2 Participants designation

Designation	Numbers
Requirement Engineer	8
System Analyst	12
Project Managers	5

4.4 Validity

Validity of research study is aimed to have the trustworthiness of results. Validity also refers that to what extent results are appropriate and subjective point of view of researchers has been biased [14]. Internal as well as external validities are maintained throughout the study. In the internal validity causal relations have been examined. Threat to internal validity has been controlled by grouping the related elements in the questionnaire. External validity is maintained by emphasizing upon the impacts of occurrence human bias, errors and activities during requirement engineering processes.

5. RESULTS AND DISCUSSION

In Table 3, average nominal score of propositions relevant to software reliability, accuracy, tolerance and estimation is determined. Availability as a sub-component of reliability is a cognitive bias. We are investigating the impacts of cognitive bias during requirement engineering processes through a RATE framework. Therefore, we have merged the cognitive biases, errors occurred during requirement engineering processes, attributes and activities performed for forecasting the damage rate. In order to see whether activities, attributes or errors and cognitive have serious impacts on software quality, we have set criteria to include or exclude the above-mentioned factors. If average nominal score remains ≥ 0.60 for a factor, it retains its position in the refined RATE framework, otherwise it is rejected.

Table. 3 Decision table from average nominal score

Component	**Sub-component**	**Average Nominal Score**	**Decision Retained in Final RATE Framework (Yes or Not)**
Reliability	Availability	0.63	Yes
	Self-descriptiveness	0.65	Yes
	Expandability	0.64	Yes
	Damaging	0.60	Yes
Accuracy	Communication Error	0.69	Yes
	Misinterpretation of Requirements	0.69	Yes
	Anchoring	0.63	Yes
	Precision	0.64	Yes
Tolerance	Cognitive Concerns	0.68	Yes
	Human Bias	0.65	Yes
	Software Size Perception	0.69	Yes

	Input Neutralization	0.64	Yes
Estimation	Forecasting	0.68	Yes
	Socio-Technical Interventions	0.67	Yes
	Judgment	0.67	Yes
	Correct Decision Making	0.63	Yes

5.1 Discussion

Reliability as a component of framework has availability, self-descriptiveness, expandability and damaging sub-components. The focus has been made to investigate the relevancy of availability bias to reliability component. From results in Table 3, we can see that availability that is a cognitive bias also works as software availability. Upon the average nominal score of all sub-components, we can say that the propositions made about each of attribute, activity performed, cognitive bias and errors are true. Our claims of truth about the propositions are supported by the opinions, views and exact answers on a scale from requirement engineers, project managers and system analysts. Software reliability is ensured when requirements are self-descriptive. Expandability as another feature of reliability determines the success of a software when new extensions are made in the existing functions of a system. Software damage rate determination impacts the future expandability of systems' requirements.

From results, it has become clear that all four sub-components of Reliability component remain the essential part of RATE Framework, because four sub-components determine the software reliability. Sequence shown in the Figure 1 determines the interaction between cognitive bias, activities, errors and software quality attributes. For example, damaging is a risk produced from errors during requirement engineering processes. Requirement engineers, system analysts and project managers present their observation in real time projects. From results, it has become clear that a correlation exists between cognitive bias and software damage rate. Software damaging is a big issue that can be only avoided if software reliability is ensured through its non-functional features like self-descriptiveness and expandability.

Accuracy component of RATE Framework encompasses the communication error, misinterpretation of requirements, anchoring and precision. Software damaging is produced from impacts of human factors. It requires time to solve the issues emerged from poor communication. If communication error is not handled properly then poor quality requirements become the requirement document. In response to proposition "Does poor communication during requirement engineering processes produce the poor-quality requirements?" the nominal score is found to be 0.72. It confirms that study participants are seriously concerned about communication errors. In another proposition "Does Misinterpretation of requirements lead towards the software failure?" the nominal score is 0.75 that too shows concerns of requirement engineers, system analysts and project managers. However, requirement engineers must be conscious about anchoring cognitive. In response to proposition "Does Anchoring bias impact negatively to success of software?" we have achieved 0.55 that shows low impact of anchoring on software accuracy. On this individual proposition, we cannot exclude the anchoring bias because average nominal score 0.63 is found. The average nominal score has been taken from Appendix A. Precision is attained when preceding two errors and an anchoring bias are eliminated.

Misjudge or misconception of requirements results into elicitation of low quality requirements. Sometimes, software elicitation personally from client cannot reveal the exact required functionalities of a system. Clients do not have sufficient knowledge about a system. It results into software failure. Software success is also impacted from the anchoring bias. In addition to anchoring bias, confirmation bias also impacts on the software quality. Anchoring and confirmation bias are taken as input for precision estimation.

Fault tolerance is an important non-functional requirement (NFR) of a system. It has been included as a main component of RATE Framework. Cognitive concerns, human bias and software size perception are cognitive bias and input neutralization acts as a software quality attribute. Fault tolerance is a non-functional requirement of a system that is derived from requirement engineering processes. Human bias directly impacts on fault tolerance feature of a system. Software size perception produces the barrier for fault tolerance estimation of a system. Software development teams are concerned over the misperception of software size. Sometimes, requirement engineers recall their past experience when they come across the misconception of a software size. However, past

experience does not always produce the accurate and complete software estimation as it is tainted by human psychological effects. Now cognitive bias is focused to find the system neutralization from extreme inputs. Impacts of earlier three cognitive biases of Tolerance component of RATE framework are totally eliminated by the execution of input neutralization as a solution to make the system more faults tolerant. Among three cognitive biases, software size misconception is more emphasized because it is commonly used for measuring the software size in software development. However, cognitive concerns and human bias are also found to be impacting factors on software tolerance from average nominal score.

Software damage rate estimation is a core of this research study that relates with other components of RATE Framework. Reliability, accuracy and tolerance as Non-Functional requirements are measured at the earlier phase of requirement engineering. Estimation as a component of RATE Framework has four sub-components such as forecasting, socio-technical interventions, judgment and correct decision making. Under the forecasting sub-component, implementation of software damage rate forecasting is checked whether it avoids or not rework on the software application development.

Forecasting becomes the input for sub-component socio-technical interventions. If cognitive concerns add the problems for accurate software damage rate forecasting, then socio-technical interventions may solve the issue of low quality requirements. It is known from results that socio-technical interventions can ameliorate the human factors, which have adverse impacts if not prevented. Human bias factors produce the errors and social-technical interventions that propagate during requirement engineering processes and diffuse these impacts of human bias factors. Judgment or perception is a human factor that impacts on the software forecasting. Better judgment can provide the accurate forecasting of software success. Requirement analysis and validation as key requirement engineering processes if handled through correct judgment can remove the errors produced in early phase of software development life cycle. The proposition "Do Requirement engineers make correct decisions to produce the better requirements for designing and coding?" is answered with the nominal score of 0.69 that signifies the importance of correct decision making to forecast the software damage rate.

5.2 Hypothesis acceptance

From results and above given analysis have shown that hypothesis H1 is accepted while H0 hypothesis is rejected. Acceptance of hypothesis H1 clearly indicates that cognitive bias, errors and activities measured during the requirement engineering processes have influences over the quality of software that in turn determines the software damage rate during requirement engineering processes.

6. CONCLUSIONS AND FUTURE WORK

Research is aimed to develop the RATE framework for the impacts of cognitive bias during the software requirement engineering processes. It is also aimed to see that how cognitive biases relate with reliability, accuracy, tolerance and estimation components of RATE framework. Several propositions about software estimation impacted from cognitive biases are presented. Cognitive bias including the confirmation, anchoring, adjustment, solution first bias and availability have been searched out from previous studies. It is also searched out that how these cognitive biases are connected with cognitive system models. Mental models and Naïve Bayes Classifier are also seen with the cognitive bias impacts. Area of cognitive bias impacts on the early phase of software development has remained a gap for earlier research works. Designing of RATE framework is described with detailed information with its components and sub-components. Reliability, accuracy and tolerance as non-functional requirements are further composed from cognitive bias and errors produced during requirement engineering processes. Cognitive bias impacts and errors are measured to forecast the success or failure of software. Average nominal scoring of all sixteen sub-components is determined to see their inclusion or exclusion from the refined RATE framework.

In future work, levels of sub-components for RATE framework can be worked out. Impact level of cognitive bias is also a future research area that can be worked out to see the low level middle level and higher level impacting on cognitive bias, errors and activities.

ACKNOWLEDGEMENT

Special thanks to Abasyn University Islamabad Campus Pakistan and Army Public College of Management & Sciences Rawalpindi, Pakistan for providing support in order to complete this research.

REFERENCES

1. Neufelder, A. M. (2010). *Ensuring Software Reliability*. New York: Marcel Deckker Inc.
2. Wilkinson, N., &Klaes, M. (2012). *An Introduction to Behavioral Economics*. New York, USA: Palgrave Macmillan.
3. Ralph, P. (2011). Toward a Theory of Debiasing Software Development. *Lecture Notes in Business Information Processing 93*, 92-105.
4. Calikli, G. (2013). Influence of confirmation biases of developers on software quality: an empirical study. *Journal Software Quality Control, 21*(2), 377-216.
5. Zhang, Y., Bellamy, R. K. E., & Kellogg, W. A. (2015). *Designing Information for Remediating Cognitive Biases in Decision-Making.* Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Seoul, Republic of Korea.
6. Gonzalez-Carrascoa, I., Colomo-Palaciosa, R., Lopez-Cuadradoa, J. L., &Penalvo, F. J. G. (2012). International Journal of Computational Intelligence Systems. *International Journal of Computational Intelligence Systems, 5*(4), 679-699.
7. Liu, J., Wang, K., Xu, A., & Wang, H. (2013). *The Analysis of Common Cause Failure based on impact vector considering human factor diversity*. Paper presented at the Industrial Electronics and Applications (ICIEA), 2013 8th IEEE Conference Melbourne, VIC.
8. Jorgensen, M., &Grimstad, S. (2012). Software Development Estimation Biases:The Role of Interdependence. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, 38*(3), 677-693.
9. Mori, T., S. Tamura, and S. Kakui. *Incremental Estimation of Project Failure Risk with Naïve Bayes Classifier*. in*Empirical Software Engineering and Measurement, 2013 ACM / IEEE International Symposium*. 2013. Baltimore, MD: IEEE.
10. Schneider, K., Liskin, O., Paulsen, H., & Kauffeld, S. (2013). *Requirements compliance as a measure of project success*. Paper presented at the Global Engineering Education Conference (EDUCON), Berlin.
11. Mohanani, R., Ralph, P., &Shreeve, B. (2014). *Requirements Fixation*. Paper presented at the International Conference on Software Engineering 14, Hyderabad India.
12. Colomo-Palacios, R., Casado-Lumbreras, C., Soto-Acosta, P., &García-Crespo, A. (2011). Using the Affect Grid to Measure Emotions in Software Requirements Engineering *Journal of Universal Computer Science, 17*(9), 1281-1298.
13. Marnewick, A., Pretorius, J.-H., & Pretorius, L. (2011). *A Perspective on Human Factors Contributing to Quality Requirements: a Crosscase Analysis.* Paper presented at the Industrial Engineering and Engineering Management (IEEM), 2011 IEEE International Conference, Singapore.
14. Runeson P., & Host, M. (2009). Guidelines for conducting and reporting case study research in software engineering, Empir Software Eng 14, 131-164.

AUTHOR PROFILE

Muhammad Hassnain completed his MS Degree in Computer Science from Abasyn University Islamabad Campus Pakistan. He received his MCS from PMAS University of Arid Agriculture Rawalpindi. Currently, he is working as a Lecturer Software Engineering at Army Public College of Management and Sciences, Rawalpindi Pakistan. His research interests are in software engineering, requirements engineering, software testing and quality assurance.

Dr. Imran Ghani is a Senior Lecturer at Monash University Malaysia Campus Malaysia. He received his Master of Information Technology degree from UAAR (Pakistan), M.Sc Computer Science from UTM (Malaysia) and Ph.D. from Kookmin University (South Korea). He his research interests are in software architecture and design, software engineering, secure software development, software application development using agile software development methods.

Appendix A: Nominal scoring

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	Total Points	Total Score	Nominal Score=Total Score/Total Points
1.	Designations of Participants							
2.	Experience in number of years							
3.	1x1=1	8x2=16	9x3=27	3x4=12	2x5=10	23x5=115	1+16+27+12+10=6	66/115=0.57
4.	0x1=0	7x2=14	7x3=21	9x4=36	1x5=5	24x5=120	0+14+21+36+5=76	76/120=0.63
5.	2x1=2	2x2=4	7x3=21	9x4=36	3x5=15	23x5=115	2+4+21+36+15=78	78/115=0.68
6.	2x1=2	2x2=4	9x3=27	5x4=20	4x5=20	22x5=110	2+4+27+20+20=73	73/110=0.66
7.	1x2=2	5x2=10	7x3=21	6x4=24	4x5=20	23x5=115	2+10+21+24+20=7	77/115=0.67
8.	1x1=1	5x2=10	9x3=27	6x4=24	2x5=10	23x5=115	1+10+27+24+10=7	72/115=0.63
9.	0x1=0	7x2=14	5x3=15	11x4=44	1x5=5	24x5=120	0+14+15+44+5=78	78/120=0.65
10.	3x1=3	1x2=2	6x3=18	9x4=36	1x5=5	20x5=100	3+2+18+36+5=64	64/100=0.64
11.	1x1=1	4x2=8	6x3=18	7x4=28	1x5=5	19x5=95	1+8+18+28+5=60	60/95=0.63
12.	0x1=0	5x2=10	8x3=24	8x4=32	2x5=10	23x5=115	0+10+24+32+10=6	66/115=0.57
13.	1x1=1	10x2=20	5x3=15	2x4=8	4x5=20	22x5=110	1+20+15+8+20=64	64/110=0.58
14.	1x1=1	4x2=8	9x3=27	7x4=28	3x5=15	24x5=120	1+8+27+28+15=79	79/120=0.66
15.	1x1=1	5x2=10	8x3=24	4x4=16	5x5=25	23x5=115	1+10+24+16+25=7	76/115=0.66
16.	1x1=1	3x2=6	3x3=9	9x4=36	5x5=25	21x5=105	1+6+9+36+25=77	77/105=0.73
17.	0x1=0	4x2=8	6x3=18	7x4=28	5x5=25	22x5=110	0+8+18+28+25=79	79/110=0.72
18.	1x1=1	4x2=8	5x3=15	8x4=32	3x5=15	21x5=105	1+8+15+32+15=71	71/105=0.68
19.	1x1=1	5x2=10	5x3=15	7x4=28	4x5=20	22x5=110	1+10+15+28+20=7	74/110=0.67
20.	2x1=2	3x2=6	5x3=15	12x4=48	1x5=5	23x5=115	2+6+15+48+5=76	76/115=0.66
21.	1x1=1	6x2=12	7x3=21	2x4=8	7x5=35	23x5=115	1+12+21+8+35=77	77/115=0.70
22.	0x1=0	3x2=6	5x3=15	9x4=36	5x5=25	22x5=110	0+6+15+36+25=82	82/110=0.75
23.	2x1=2	5x2=10	6x3=18	5x4=20	4x5=20	22x5=110	2+10+18+20+20=7	70/110=0.64
24.	3x1=3	2x2=4	8x3=24	5x4=20	4x5=20	22x5=110	3+4+24+20+20=71	71/110=0.65
25.	1x1=1	4x2=8	2x3=6	12x3=36	2x5=10	21x5=105	1+8+6+36+10=61	61/105=0.58
26.	2x1=2	4x2=8	8x3=24	7x3=21	1x5=5	22x5=110	2+8+24+21+5=60	60/110=0.55
27.	1x1=1	2x2=4	3x3=9	12x4=48	3x5=15	21x5=105	1+4+9+48+15=77	77/105=0.73
28.	2x1=2	4x2=8	6x3=18	9x4=36	1x5=5	22x5=110	2+8+18+36+5=69	69/110=0.63
29.	1x1=1	5x2=10	7x3=21	5x4=20	3x5=15	21x5=105	1+10+21+20+15=6	67/105=0.64
30.	0x1=0	6x2=12	7x3=21	5x4=20	2x5=10	20x5=100	0+12+21+20+10=6	63/100=0.63
31.	2x1=2	4x2=8	7x3=21	6x4=24	3x5=15	22x5=110	2+8+21+24+15=70	70/110=0.64
32.	1x1=1	3x2=6	5x3=15	9x4=36	4x5=20	22x5=110	1+6+15+36+20=78	78/110=0.71
33.	0x1=0	3x2=6	8x3=24	5x4=20	5x5=25	21x5=105	1+6+24+20+25=75	75/105=0.72
34.	2x1=2	7x2=14	2x3=6	7x4=28	3x5=15	21x5=105	2+14+6+28+15=65	65/105=0.62
35.	0x1=0	6x2=12	6x3=18	8x4=32	2x5=10	22x5=110	0+12+18+32+10=7	72/110=0.65
36.	1x1=1	4x2=2	4x3=12	11x4=44	1x5=5	21x5=105	1+2+12+44+5=64	64/105=0.61
37.	2x1=2	5x2=10	8x3=24	4x4=16	3x5=15	22x5=110	2+10+24+16+15=6	67/110=0.61
38.	2x1=2	4x2=8	3x3=9	9x4=36	2x5=10	20x5=100	2+8+9+36+10=65	65/100=0.65

39.	0x1=0	2x2=4	8x3=24	8x4=32	1x5=5	19x5=95	0+4+24+32+5=65	65/95=0.68
40.	2x1=2	3x2=6	4x3=12	9x4=36	3x5=15	21x5=105	2+6+12+36+15=71	71/105=0.68
41.	1x1=1	4x2=8	6x3=18	8x4=32	4x5=20	23x5=115	1+8+18+32+20=79	79/115=0.69
42.	2x1=2	2x2=4	8x3=24	7x4=28	2x5=10	21x5=105	2+4+24+28+10=68	68/105=0.65
43.	1x1=1	3x2=6	5x3=15	9x4=36	4x5=20	22x5=110	1+6+15+36+20=78	78/110=0.71
44.	0x1=0	4x2=8	8x3=24	8x4=32	4x5=20	24x5=120	0+8+24+32+20=84	84/120=0.70
45.	1x1=1	5x2=10	7x3=21	8x4=32	1x5=5	22x5=110	1+10+21+32+5=69	69/110=0.63
46.	0x1=0	6x2=12	8x3=24	5x4=20	2x5=10	21x5=105	0+12+24+20+10=6	66/105=0.63
47.	1x1=1	3x2=6	8x3=24	8x4=32	2x5=10	22x5=110	1+6+24+32+10=73	73/110=0.66
48.	0x1=0	4x2=8	7x3=21	7x4=28	4x5=20	22x5=110	0+8+21+28+20=77	77/110=0.70
49.	2x1=2	6x2=12	4x3=12	7x4=28	4x5=20	23x5=115	2+12+12+28+20=7	74/115=0.64
50.	0x1=0	4x2=8	9x3=27	8x4=32	3x5=15	24x5=120	0+8+27+32+15=82	82/120=0.68
51.	2x1=2	4x2=8	3x3=9	9x4=36	4x5=20	22x5=110	2+8+9+36+20=75	75/110=0.68
52.	0x1=0	5x2=10	5x3=15	8x4=32	3x5=15	21x5=105	0+10+15+32+15=7	72/105=0.69
53.	3x1=3	1x2=2	7x3=21	9x4=36	2x5=10	22x5=110	3+2+21+36+10=72	72/110=0.65
54.	2x1=2	6x2=12	4x3=12	9x4=36	2x5=10	23x5=115	2+12+12+36+10=7	72/115=0.66
55.	2x1=2	4x2=8	6x3=18	7x4=28	4x5=20	23x5=115	2+8+18+28+20=76	76/115=0.66
56.	2x1=2	2x2=4	3x3=9	10x4=40	5x5=25	22x5=110	2+4+9+40+25=80	80/110=0.72
57.	2x1=2	3x2=6	7x3=21	5x4=20	3x5=15	20x5=100	2+6+21+20+15=64	64/100=0.64
58.	1x1=1	5x2=10	3x3=9	11x4=44	3x5=15	23x5=115	1+10+9+44+15=79	79/115=0.69
59.	1x1=1	6x2=12	6x3=18	9x4=36	2x5=10	24x5=120	1+12+18+36+10=6	67/120=0.56
60.	2x1=2	5x2=10	6x3=18	6x4=24	4x5=20	23x5=115	2+10+18+24+20=7	74/115=0.64

Appendix B: Survey Questionnaire

Software Damage Rate Forecasting

Dear Participants

You are invited to take part in the study on RATE framework development for forecasting the software damage rate during the requirement engineering processes. This research study is aimed to see the impacts of human factors on the success and quality of software development in Software development industry of Pakistan. It is also investigated to find the human factors, which can be analyzed to enhance the software quality through software damage rate forecasting. This research study is conducted at Abasyn University Islamabad Campus to complete my degree of MS. Computer Science. Therefore, I request you to complete the questionnaire given in the following. The provided information will be used for research purposes. Thanks use following Rating criteria as to fill the questions from 3 to 60.

1= strongly disagree, 2= disagree, 3= neutral, 4=agree. 5= strongly agree

Availability
A sub component of Reliability

Reliability
A component of RATE Framework

Does availability (recalling the information or disseminate from that just happened) heuristic impacts the software damage rate?
Does availability bias remain consistent with all types of systems' damage rate estimation?
Does availability exist in anchoring of human issues?

Self-Descriptiveness
A sub component of Reliability

Does measurement of efforts for self descriptiveness support the software success or failure rate estimation?
Does inadequate information have perceptual issues for a client that lacks understanding about software expansion in future?
Does Self descriptiveness always supersede other software reliability features like consistency, correctness and completeness etc?

Expandability
A sub component of Reliability

Do requirements expand-ability influences the software estimation?
Does software damage rate estimation provide the right direction for requirements expand-ability in future?
If requirement expand-ability is misjudged, then client suffers more than judgment authority?

Damaging
A sub component of Reliability

Does a correlation between cognitive bias and software damage rate exit?
Is it right that software damage rate can be determined through software requirement specification?
Is software damage rate estimation from human factors a reliable forecasting?

ACCURACY
A component of RATE Framework

Communication Error
A sub-component of ACCURACY

Do requirement changes result from poor communication between system stakeholders?
Does Poor communication between client and requirement engineer impact the software estimation?
Does poor communication during requirement engineering processes produce the poor-quality requirements?
Does communication error produce the delayed software requirement specification?
Does communication involve the human process that is not solved through the engineering solution?
Does software damage result from communication error between Requirement Engineers and clients?

Misinterpretation of Requirements
A sub-component of ACCURACY

Does requirement gathering phase provide accurate software damage rate estimation?
Does Misinterpretation of requirements lead towards the software failure?
Do good development teams develop good projects through better communication?

Anchoring
A sub-component of ACCURACY

Does anchoring bias produces less accuracy in software damage rate estimation?
Does Anchoring as information processing has close relation with mental simulation and story generation?
Does Anchoring bias impact negatively to success of software?
Confirmation bias occurring at requirement engineering processes impacts the software quality?

Precision
A sub-component of ACCURACY

Does anchoring and adjustment bias prevent from making precise estimation?
Is Precision an important software quality measuring parameter?
Is software accuracy highly derived from precision in software success/failure rate estimation?
Does software accuracy depend upon the precision of software estimation?

Tolerance
A component of RATE Framework

Cognitive Concerns
A sub-component of Tolerance Component

Does fault tolerance of software derive from requirement engineering processes?
Are cognitive concerns early perceived increase the level of system tolerance?

Is it necessary for requirement specification to be tolerant of the incompleteness?
Do cognitive concerns of software impact the human judgment about software damage rate forecasting?

Human Bias
A sub-component of Tolerance Component

Human bias produces requirement inconsistencies, which reduce the fault tolerance level of software?
Is human bias free of software fault tolerance more vital than technical faults?
Does Requirement Engineer and Client face the challenge of judgment bias that increases the system in-tolerance?
Reliable requirements come from understanding of stakeholders' emotions.
Emotions as Human behavior perception and judgment affect the software quality.

Software Size Perception
A sub-component of Tolerance Component

Does software size produce the barrier for system fault tolerance estimation?
Mis-perception of software size produces the concerns for development team?
Does sometimes self-perception enforce to recall the past experience?
Recalling the past experience that can be incomplete and inaccurate is also tainted by the psychological effects?

Input Neutralization
A sub-component of Tolerance Component

Does cognitive bias neutralize the system from extreme inputs, which can tolerate the impacts of extreme inputs?
Solution-first bias leads the system's stakeholders to frame a solution for those human cognitive issues which they cannot understand?
Does confirmation bias aggravate the solution first bias that leads towards the poor software design decisions?

Estimation
A component of RATE Framework

Forecasting
A sub-component of Estimation Component

Does cognitive based software estimation increase the efficiency of software damage rate estimation?
Do you think that decision makers apply the information judgment to ensure the better software damage rate forecasting?
Is damage rate forecasting implemented to avoid the rework on software applications?
Is software failure/success forecasting more reliable at early phases of software development?

Socio-Technical Interventions
A sub-component of Estimation Component

Do socio-technical interventions, which ameliorate the impacts of human factors and resulting faults, increase the software quality?
Interventions can be propagated during software requirement engineering processes to overcome the errors produced from biasing factors?
Do socio-technical interventions require training to avoid the social issues and get skilled?

Judgment
A sub-component of Estimation Component

Is judgment a natural phenomenon or attained through experience in software industry?
Does better judgment always reveal an accurate forecasting of software failure rate?
Is human behavior influence its judgment capabilities in measuring the software damage rate?

Correct Decision Making
A sub-component of Estimation Component

Do Requirement engineers make correct decisions to produce the better requirements for designing and coding?
Can correct decision making helps to estimate the software damage rate in an appropriate way?
Can requirement analysis and validation through correct judgment remove the errors observed in early phase of software development life cycle?