AN EVALUATION STUDY OF GENERAL SOFTWARE PROJECT RISK BASED ON SOFTWARE PRACTITIONERS EXPERIENCES

Sahand Vahidnia[1] and Ömer Özgür Tanrıöver[2] and I. N. Askerzade[3]
[1 23]Ankara University / Computer Engineering Department, Ankara University, Gölbaşı 50.yıl Yerleşkesi Bahçelievler Mh., 06830 Ankara 06100, Turkey

ABSTRACT

Acritical process in a software project life-cycle is risk assessment and mitigation. Risks exist in every software project and recognizing and evaluating risks and uncertainties is a challenging process for practitioners with little historical data. In our study, by using a survey data, we identify and provide a relatively wider coverage of risks and ratings of software project. The risk register and evaluations are useful for practitioner of small organizations at initial phase of risk identification and assessment. There are 128 risks in this study which are analyzed. Furthermore, the study provides a top risk list according to this study alongside a highly cross correlatedrisks table. Additionally, previous studies have also provided top risks lists regarding the corresponding surveys. This study extends previous studies to provide a recent risk study. Outcomes are also compared to previous works in discussion.

KEYWORDS

Software Project Risk Assessment, Risk Matrix, Software Engineering

1. INTRODUCTION

As indicated by researches [1] [2][3], the worldwide software business sector is evaluated to have an estimation of US$330 billion in 2014. Further, the Chaos Report from the Standish Group reported that the success rate of worldwide (fundamentally U.S. and Europe) software projects in 2015 is only 29% [4].In all projects, risks should be identified, assessed by its probability of occurrence and impact, and a contingency plan should be developed for remediating the problem actually occur [5]. The procedure of risk identification can be troublesome and misleading in some cases. Moreover, knowing whether a risk is important or not is also is another issue which requires more research on cases.

This research`s aim is to assess the initial rating of risks for small to medium size software projects. One can see that in recent years, there are not as much studies to determine general risk factors and their possible effects on small to medium size software projects. In order to identify general risks and deploy a survey and obtain rating from practitioners, previous researches in risk assessment and management field had to be reviewed. Some of these studies are briefly given below.

Zardari[6] in his research about managing software risks, attempts to provide practitioners with necessary basic information regarding the management of risks in software. The research defines the term of proactive and reactive management, a list of top risks, probability and impact and their levels alongside other terms.

Keshlaf[7] specifically focuses on web and distributed software development project risks in this study, and provides a summarized risk set from other works. The research studies risks and challenges in the field and also studies and analysis existing methods and frameworks of software

DOI:10.5121/ijcsit.2016.8601

risk management including SD-RM-Concept and EBIOS Methodology.

One of the most influential works in the field is Arnuphaptrairong`s research [8] regarding the risk factor lists from other works and top risks of each list. The research compares and analyzes top risk lists from Bohem and other studies. As for our study, we had a comparison of results amongst the top risks and our research results. Further assessment is conducted in discussion section.

Song's research [9] is another research which attempts to show the importance of risk management in software engineering projects. The research utilizes an information entropy approach to analyze risks in three dimensions of loss, uncertainty and probability. The research also provides awide risk list which was directly used in this research.

In other studies including [10] risk avoidance models are provided and some researches like [11] have gathered and explained and analyzed risks. Some studies have taken different approaches like analyzing multi characteristics of risks like multi dimension and multi situations and etc. like Wang's work [12].

One of the problems with previous studies is that each adopt a sets of risk factors which are sometimes completely disjoint or overlapping. Also most of the studies consider a limited set of risks, therefore we aim to put a wide coverage list of software related risks. We ask software practitioners for their experience over of these risks and study the result to generate a super set of risks and ratings for reference.

In our study, by conducting a survey, we mean to distinguish and provide a wider coverage of initial rating to software general project risks and their relations especially for practitioner of small organizations. A top risk list, highly cross correlated risks are provided for use by software teams or practitioners with relatively little experience and previous historical data. The identified wider coverage of general risks and ratings may be used in further studies. Furthermore, we discuss the outcomes and compare our results to other results from a survey of previous works[8].

2. GENERAL SOFTWARE PROJECT RISK FACTOR COLLECTION

Although, in previous studies diverse risk factors, categories and analysis tools have been utilized, in this study we considered studies using keywords of "risk" and "software". Some of the studies have a wider coverage over risks. Among cited researches, throughout recent years, especially from 2006 to 2016, there have been few researches about software development risk factors. However, since 2006, many aspects of software engineering have evolved.

In various studies, risk factors included are constrained to particular aspect or phase of software development. But it is desired to have a near complete set of general risk factors and their effects. So as the initial phase of this study, we accumulated risk factors and categories from related researches [6], [7],[9],[10], [11], [13], [14], [15]. By gathering risk factors, we came up with a superset of risks which has a greater coverage over risks of software projects and development phases. But the risk set, in addition to have too many risks to consider and assess, includes numerous similar and covering risks. So an action is taken to minimize the risk set.

In order to discover similar risk factors, at first a keyword search was performed. This search prompted a keyword sorting, and also finding exceedingly comparative risks which are dispensed based on their definitions. Additionally, some available risks were unpredictable and were outcome of project rather than a possible risk to endanger project. These risks are also eliminated from the list. Next table is presenting few examples of eliminated risks.

Table 1- Risk elimination example

Development Over-Schedule	Team	Disaster Recovery
We consider over schedule development as a failure type, rather than a risk. Over scheduling is result of many risks happening during project design and development phases.	Team, itself is too general to be considered as a risk factors. There are other risk factors in this study which will cover team.	The risk of bad disaster recovery is important, but we decided to have the risk of Disaster / Catastrophe to cover this risk and prevent overlapping.

The corresponding table of risk-factors and gathered data is shown in Appendix. in its final state. This phase led us to a set of risk factors which itself can be helpful for development teams as initial set for risk register formulation.

To make it possible for survey participants to judge and evaluate the risk factors, we reworded the risk factors in a way that these factors are consistent, understandable and rated more naturally. For instance, “Project Management Approach / method” has a high risk potential and possibly will receive a 2 or 3 risk rating from experts (moderate to high). But for a developer it's difficult to evaluate a project considering this factor. In order to address this issue, we added "wrong method" to this factor in parenthesis. In short we make factors easier to be rated, by adding adjectives or other words to factor phrases. Other examples:

- Definition of the Program (ambiguity)
- Mix Of Team Skills (Bad Or Low)
- Technical Feedback (lack of)

Furthermore, all risk factors are stated to with negative statements. As an example "programming language experience" factor is actually a positive and good factor in project. So listing this in risk factors may confuse experts and developers. So, we added "lack of" in parenthesis to risk factor. Now probability of confusion for participants is less likely to occur.

Risk can be divided into three items of probability, impact and risk [16]. Risk is product of probability and impact items [17]. Scale definitions of probability and impact levels are reused from[6].

$$Risk\ Score = Probability * Impact$$

Probability levels definitions:

1. High / Very Likely: High chance of this risk occurring, thus becoming a problem {70% < x}.

2. Medium / Probable: Risk like this may turn into a problem once in a while {30% < x < 70%}.

3. Low / Improbable: Not much chance this will become a problem {0% < x < 30%}.

Impact levels definitions:

1. High / Catastrophic: Loss of system; unrecoverable failure of project; major problem; schedule slip causing launch date to be missed; cost overrun greater than 50% of budget.

2. Medium / Critical: Considerable problem with project with recoverable operational capacity; cost overrun exceeding 10% (but less than 50% of planned cost).

3. Low / Marginal: Minor problem project; recoverable loss of operational capacity; internal schedule slip that does not impact launch date cost overrun less than 10% of planned cost or timeframe.

Further to make risk factors more understandable and precise, we gave brief information about some risk factors from [18].

3. DATA COLLECTION AND ANALYSIS

3.1. EXPERT DATA COLLECTION

An essential phase in this research is acquisition of expert data. An onlıne survey data is used for this purpose which was conducted during the research. The purpose of this survey was to collect the risk-factor probabilities and potential impact based on real projects of developers which are happened in the past. The survey comprises 3 parts of *I*, *II* and *III*. In part *I* we asked developers to give us a brief information about an unsuccessful/challenged/failed project which they participated in. There are 12 important questions in part *I*, including questions about experience of the expert and questions about project size and type. These part also comprises questions about programming paradigm, design pattern and methodology of project. Part *II* of the survey contains 10 questions about project's fate. Questions in this part are generally excerpted from references[6], [7], [10], [13], [14] and some are adjusted to prevent the possible confusion in answering. This section provides the failure data of projects which will be evaluated on future studies. Part *III* consists of the risk factor ratings Part *III* consists of the risk factor ratings as described earlier. The analysis of part *II* is elaborated in a yet to be published study. In this one, we analyze information obtained in part *III*. The structure of the survey can be accessed at http://survey.labs.tips/result.php .

At this period of the research, assembled data is pre-processed. To do as such, all faulty data are eliminated. There were a total of 86 participants in this survey, yet some answers were inadequate and inappropriate which were removed. Also to prevent cheating (random answers) 5 check questions were among part III questions. Toward the end, there were only 40 dependable answer which are considered in final dataset.

Subsequent to making a rectified dataset, dataset is processed. The mean of all entered data for each risk is used to order risks by importance according to current survey data, considering the risk factors shown in appendix. The fundamental objective of this research is aggregation and usage of these risk-factors in a more applicable manner for real world software projects. So we chose to implement risk matrix at this stage.

3.2. GENERAL RISK MATRIX

Risk matrices are most likely one of wide spread tools for risk evaluation. Risk matrix is much easier to comprehend than raw data and other methods. They are for the most part used to determine the extent of a risk and whether or not the risk is adequately controlled. Probability and Impact are two dimensions of a risk matrix. The combination of these dimensions creates a risk

matrix which makes the assessment easier. Risk matrix dimensions or axes are divided into 3 level each, which creates a 9 cell qualitative matrix [19]. This matrix has 3 part:

- High / Major Concern (red): Risk is high in these sections and an action should be taken.
- Medium / Concern (yellow): Risk is moderate in these sections and there is a chance that risks in these areas may affect project.
- Low / No Concern (green): Risk in these sections are low and acceptable and can be ignored.

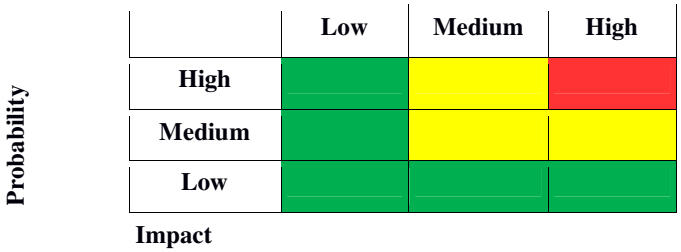

Figure 1 - Risk Matrix

There are likewise different arrangements of risk matrices like 5x5, 7x5 and 7x4 risk matrices which are not adapted in this study due to simplicity of 3x3 matrices [20].

A risk matrix is acquired utilizing averages of 40 data points for every risk factor according to Figure 1. According to this general risk matrix, only risk factors listed below are categorized as relatively important.

Table 2 – Top risk factor list according to risk matrix

#	Risk Factor	#	Risk Factor
1	Lack Of Development Technology Experience Of Project Team	11	Short Term Solution (lack Of Long Term Solution)
2	Project- Resource Conflict	12	Lack Of Testability
3	Large Project Size	13	Implementation Difficulty
4	Bad Project Management Approach / Method	14	Late Identification Of Defects
5	Lack Of Project Management Experience	15	Bad Defect Tracking
6	Poor Project Planning	16	Lack Of Experience With Software Engineering Process
7	Expansion Of Software Requirements	17	Lack Of Training Of Team
8	Low Knowledge And Understanding Of Clients Regarding The Requirements	18	Dependency On A Few Key People
9	Bad Development Schedule	19	Need To Integrate With Other Systems
10	Lack Of Analyst Capability	20	Lack Of Platform Experience

Risk matrix take into account both probability and impact perspective. Analysis and comparisons regarding the table 2 is discussed at discussion section.

3.3. FREQUENCY AND CORRELATIONS

For the 128 risk factors and 40 data points, a statistical preprocessing is conducted. Firstly, descriptive statistics and frequencies has been obtained for results to achieve a better understanding over data points. Calculating average risk scores for available data points is

required in this step. In order to perform this action, two methods could be implemented. The first method is simply calculating products of acquired average impact and probability values. This approach yields in loss of validity because of ordinal scale definition. Hence, to produce risk score from multiplication of each probability and impact data point, and then calculating the average and frequencies.

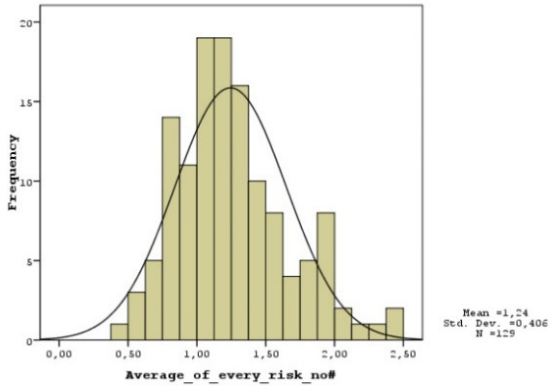

Figure 2 – Frequency of average of every risk

Table 3 – Frequency data

	Probability	**Impact**	**Risk Val.**
Risk No.	128	128	128
Std. Deviation	0.19815	0.22983	0.40558
Std. Error of Mean	0.01745	0.02024	0.03571
Range	1.00	1.30	2.03
Minimum	0.10	0.35	0.43
Maximum	1.40	1.65	2.45
Mean	0.8587	0.8965	1.244

For average of the 40 data points of every one of the128 risk factors, rest of calculated data are appeared in the table above. As exhibited in figures above, most risks have an average of 1.2 which is the reason average data is not a decent information to rely on. So we decided to analyze data using frequencies. When we investigate frequencies of risks, rather than comparing their averages, we compare the number of people who marked a risk as important. By important we mean a risk score of 6 or 9. To do so, all scores except 6 and 9 among all data are excluded from evaluation to compare risks with dangerous behavior only. Table 5 also demonstrates the generated top risk list.

Another objective of this study was to determine fundamental risk factors on software project failures. In order to see if one consider only subset of risk factors which have high effect on project failures, a dimensionality reduction technique was considered.

To minimize the risk factor dimension correlation coefficients of 128 risk factors are extracted. Since the risk data is distributed normally, Pearson's correlation coefficient is used as a statistical measure of the strength of a linear relationship between paired data. In a sample it is denoted by r and is by design constrained as follow:

$$-1 \leq r \leq +1$$

According to Weinberg [21], Pearson correlation coefficients of $r = \pm 0.5$ are considered strong and correlation coefficients close to ± 1 are the strongest. Also 0 means there are no correlation among variables. Evans [22] recommends a correlation coefficient of ± 0.6 *to* ± 0.79 as a strong and ± 0.8 *to* ± 1*as very strong* correlation coefficient. In our calculation, correlation coefficients which are among ± 0.8 *and* ± 1 are considered as very strong. Next table shows a list of very strongly cross correlated risk factors i.e± 0.8. Strong cross correlation among risk pairs creates a duplicate variable effect which makes the data unhealthy.

Statistical calculations in this research was performed using SPSS tool [23], except for the correlation values which are generated using Matlab's Pearson's correlation function.

Table 4 - Very strongly correlated risks

Very Strongly Correlated			**Some Highly Correlated (among ±0.7 and ±0.8)**					
Risk ID	Risk ID	Correlation Coefficient	Risk ID	Risk ID	Correlation Coefficient	Risk ID	Risk ID	Correlation Coefficient
18	**125**	***0.83277***	3	126	*0.73161*	65	120	*0.76273*
55	**111**	***0.84347***	9	54	*0.70722*	70	74	*0.74817*
37	**51**	***0.82162***	26	11	*0.71208*	71	93	*0.77567*
21	**30**	***0.80628***	27	48	*0.71146*	97	115	*0.75964*
87	**93**	***0.85617***	28	95	*0.72975*	84	98	*0.70398*
24	**50**	***0.85583***	29	31	*0.77262*	85	120	*0.74108*
25	**74**	***0.84403***	34	109	*0.70646*	105	124	*0.70634*
90	**109**	***0.80085***	47	126	*0.71017*	122	115	*0.76228*
38	**126**	***0.84679***	56	74	*0.78317*	43	104	*0.71679*
101	**111**	***0.80295***	57	88	*0.74041*	44	95	*0.71029*
39	**49**	***0.82401***	58	117	*0.70219*	36	126	*0.74008*

There are a total of 33 highly correlated risks separated from entire risk factor list. Statistically these 33 risks, represent rehashed data among 128 risk. Due to limits in gathering data-points and low confidence in data analysis, it is not encouraged to rely on this correlation data for eliminating all 33 risk factors.

Furthermore, we investigated few risks in correlation table and discovered that some correlations' logically make sense and some don't. For instance, risks of "Poor Project Planning" and "Dependency On a Few Key People" are highly correlated with a score of 0.80 and there is a real logical relation visible to engineers. On the other hand, there are no direct logical relation among some risks like "Short Term Solution (lack of Long Term Solution)" and "Staff Turnover" with correlation coefficient of 0.83. The issue needs to be addressed in future works after acquiring more project and expert data.

4. DISCUSSION

As mentioned previously, [8] has conducted a literature review to compare risk factors in other studies. In this study we also provide two tables of top risks with different methods. In this section combination of top risks provided in [8] is compared to our results. Due to differences in definition and implication of risks introduced and compared in other studies, we consider closest risk and overlapping risks in definition and mark them with a star sign (*). It is worth mentioning that prior to elimination of overlapping risks in section II., many overlapping risks were in our superset of risks. This partially justifies our use overlapping and similar risks in table 5.

Table 5 – Comparison table of current and previous researches

#	TOP 20 RISKS (WITH SUM OF OVER 30)	SUM	A	B	C	D	E	F	G	H	I
1.	Development Technology Not Match To Project	30		*						+	
2.	Lack Of Analyst Capability	30	*			+		+	+	+	
3.	Short Term Solution	30	*								
4.	Lack Of Organizational Maturity	30									*
5.	Large Project Size	33	*								
6.	Lack Of Training Of Team	33	*	*			*	*			*
7.	Lack Of Experience With Software Engineering Process	33	*								
8.	High System Dependencies	33									
9.	Problem With Hardware Platform	33									
10.	Lack Of Project Management Experience	36	*		*		*	*			*
11.	Incorrect Project Size Estimation	36		*				*			
12.	Dependency On A Few Key People	36									
13.	Lack Of Or Bad Change Control For Work Products	36			*		*				
14.	Large Database Size	39	*								
15.	Bad Development Schedule	39		*				*			
16.	Lack Of Platform Experience	39	*	*				*			
17.	Expansion Of Software Requirements	42	*	*				*		*	
18.	Bad Project Management Approach / Method	42	*		*			*		*	
19.	Lack Of Use Of Modern Programming Practices	42								*	
20.	Lack Of Reusable Components	42									

A. Is available at table 2
B. Is available at top 10 list of Bohem[8]
C. Is available at top 10 list of Schmidt et al. (USA) [8]
D. Is available at top 10 list of Schmidt et al. (HONG KONG) [8]
E. Is available at top 10 list of Schmidt et al. (FINLAND) [8]
F. Is available at top 10 list of Addison and Vallabh[8]
G. Is available at top 10 list of Addison [8]
H. Is available at top 10 list of Han and Huang [8]
I. Is available at top 10 list of Pare et al. [8]

+ means the risk can also cover named risk, or the risk can be concluded from named risk, but not precisely the same.

** means the risk is very similar or exactly the same*

10 risk factors out of 20 risk factors in table 5 are also present in table 2. Table 2 was generated using risk matrix, but due to general project risks, unless the data is not filtered into sub categories, arithmetic mean on data will be misleading. As a result, the data presented in table 5 is much healthier and reliable.

As presented in table 5, many of the important risks from other works, are not recognized as important in our survey data. Also the reason might not be clear in some cases, but in many cases it can be explained. We suggest that these differences in the results can be related to time dimensions of conducted researches regarding the risks. Also [8] is suggesting a similar explanation for cases like Bohem. These researches are conducted between the years of 1991 and

2008 which makes our hypothesis stronger. In past two decades, technology advancements have had great contributions to advancements and changes in software industry and software risk factors and their effects. As an example, “Developing wrong user interface” might not be that much of a concern for software developers nowadays.

Another finding in this comparison is that, unsurprisingly our survey participants did not recognize user related risks as important. This might indicate the lack of management experience, dealing with clients and users for the participants, as our survey participants are all composed of young computer engineers, students and software developers.

Another proof for this finding is that the risk of “Lack of Reusable Components” has not been repeated or slightly mentioned in prior researches, whereas in this survey, this risk is on top of the list. This indicates that participants were too much concerned about programming than managing and user dimension.

5. CONCLUSION

In this study we introduce a reference software risk set for software development. We also provide a risk matrix for software projects which can be used to assess probable and common risks in developing software solutions.

Also in this study a dimensionality reduction was performed using statistical correlations among risk factors. So software practitioners can see correlations of risks as Table 4 and predict which risk may also affect them according to previous risks.

Throughout this study analysis on survey data was conducted to highlight important risks in software project development, using risk matrix and a custom method of priority table. For better precision, further studies and more data-points are required, which will be available in next research. Having more data-points in addition to having more reliable risk matrix and risk priority tables, will also make it possible to perform other analysis including regression analysis.

ACKNOWLEDGEMENTS

The authors would like to thank all participants in the survey for their great contributions to this research.

REFERENCES

[1] Hu, Yong, Zhang, Xiangzhou, Ngai, E.W.T., Cai, Ruichu ,Liu, Mei, "Software project risk analysis using Bayesian networks with causality constraints," Decision Support Systems, pp. 439-449, 2013.
[2] "Research and Markets, Software: Global Industry Guide," 2010.
[3] T. S. Group, "The Chaos Manifesto," The Standish Group International, Incorporated, 2013.
[4] "CHAOS Report 2015," The Standish Group International, 2015.
[5] R. S. Pressman, Software Engineering, A Practitioner's Approach, 7, Ed., 2010.
[6] S. Zardari, "Software Risk Management," in 2009 International Conference on Information Management and Engineering, 2009.
[7] Ayad Ali Keshlaf,Steve Riddle, "Risk Management for Web and Distributed Software Development Projects," in The Fifth International Conference on Internet Monitoring and Protection, 2010.
[8] T. Arnuphaptrairong, "Top Ten Lists of Software Project Risks: Evidence from the Literature Survey," in Proceedings of the International MultiConference of Engineers and Computer Scientists (IMECS2011), 2011.
[9] Hao Song, Dengsheng Wu, Minglu Li, Chen Cai1, Jianping Li,, "An entropy based approach for software risk assessment: A perspective of trustworthiness enhancement," Software Engineering, pp.

575-578, 2010.
[10] Shahzad Basit , Abdullah S. Al-Mudimigh, "Risk Identification, Mitigation and Avoidance Model for Handling Software Risk," in Second International Conference on Computational Intelligence, Communication Systems and Networks Risk, 2010.
[11] Shahzad Basit,Ihsan Ullah, Naveed Khan, "Software Risk Identification and Mitigation in Incremental Model," in 2009 International Conference on Information and Multimedia Technology, 2009.
[12] Yu Wang, Shun Fu, "A General Cognition to the Multi-characters of Software Risks Yu," in International Conference on Computational and Information Sciences, 2011.
[13] Li Xiaosong, Liu Shushi, Cai Wenjun, Feng Songjiang, "The Application of Risk Matrix to Software Project Risk Management," International Forum on Information Technology and Applications, pp. 480-483, 2009.
[14] Appari, Ajit, Benaroch, Michel, "Monetary pricing of software development risks: A method and empirical illustration," Journal of Systems and Software, pp. 2098-2017, 2010.
[15] Dengsheng Wu, Hao Song, Minglu Li, Chen Cai, Jianping Li, "Modeling Risk Factors Dependence Using Copula Method for Assessing Software Schedule Risk," Beijing,, 2010 .
[16] Haisjackl Christian, Felderer Michael, Breu Ruth, "RisCal - A Risk Estimation Tool for Software Engineering Purposes," in 39th Euromicro Conference Series on Software Engineering and Advanced Applications, 2013.
[17] Olid, Khan, Mannan, Bin, "A Review of Software Risk Management for Selection of best Tools and Techniques," in Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing,, 2008.
[18] Haneen Hijazi,Shihadeh Alqrainy,Hasan Muaidi,Thair Khdour, "Risk Factors in Software Development Phases," European Scientific Journal, vol. 10, no. 3, pp. 213-232, 2014.
[19] Gary Stoneburner, Alice Goguen, Alexis Feringa, "Risk Management Guide for Information Technology Systems," National Institute of Standards and Technology, no. sp800-30, 2002.
[20] Adam S. Markowski, M. Sam Mannan, "Fuzzy risk matrix," Journal of Hazardous Materials, vol. 159, no. 1, pp. 152-157, 2008.
[21] Sharon Lawner Weinberg, Sarah Knapp Abramowitz, Statistics Using SPSS: An Integrative Approach, Cambridge University Press, 2008.
[22] J. D. Evans, Straightforward Statistics for the Behavioral Sciences, Brooks/Cole Publishing Company, 1996.
[23] IBM, "IBM SPSS Software," IBM Corporation, 2016. [Online]. Available: http://www.ibm.com/analytics/us/en/technology/spss/. [Accessed Dec 2015].
[24] Zhiwei Xu, Taghi M. Khoshgoftaar, Edward B. Allen Motorola, "Application of fuzzy expert systems in assessing operational risk of software," Information and Software Technology 45, pp. 373-388, 2003.

APPENDIX

Unsorted Risk Statement	**Medium / Critical Impact**	**Probable / Medium probability**	**Reference**	**Unsorted Risk Statement**	**Medium / Critical Impact**	**Probable / Medium probability**	**Reference**
1. Large Database Size	*	*	[14]	65. Developing Wrong Software Functions			[6][7]
2. Main Storage Constraint			[14]	66. Developing Wrong User Interface			[6][7]
3. High Platform Volatility			[14]	67. Gold Plating			[6][7]

4. Bad Development Schedule			[14]	68. Shortfalls In Outsourced Components			[6][7]
5. Lack Of Analyst Capability	*	*	[14]	69. Shortfalls In Externally Performed Tasks			[6][7]
6. Lack Of Platform Experience	*	*	[14]	70. Real-time Performance Shortfalls			[6][7]
7. Lack Of Use Of Modern Programming Practices		*	[14]	71. Bad Traceability			[7]
8. Low Usage Of Software Support Tools		*	[14]	72. Insufficient Verification And Validation			[7]
9. Lack Of Software Developer Competence			[24]	73. Customer Unsatisfied At Project Delivery			[7]
10. Project NOT Fit To Customer Organization			[13]	74. Risk Reducing Technique Producing New Risk			[7]
11. Lack Of Customer Perception			[13]	75. Catastrophe / Disaster			[7]
12. Project- Resource Conflict	*	*	[13]	76. Incorrect Project Size Estimation	*		[11]
13. Customer Conflict			[13]	77. Project Funding Uncertainty	*	*	[11]
14. Lack Of Leadership			[13]	78. Rapid Change Of Job			[11]
15. Definition Of The Program (ambiguity)			[13]	79. Change In Working Circumstances By Management			[11]
16. High Political Influences			[13]	80. Hardware Default Changes			[11]
17. Inconvenient Date			[13]	81. Requirement Postponement		*	[11]
18. Short Term Solution (lack Of Long Term Solution)	*	*	[13]	82. Presence Of High Bugs/errors Count			[11]
19. Lack Of Organization Stability			[13]	83. Technology Change			[11]
20. Lack Of Organization Roles And Responsibilities			[13]	84. Underestimation Of Data Increase Due To Software Success			[11]
21. Lack Of Policies And Standards			[13]	85. Lack Of Design And Development Tool Independence			[11]
22. Lack Of Management Support And Involvement			[13]	86. Risk Of Intruders (hackers, Viruses, Trojan Horse)			[11]
23. Lack Of Project Objectives		*	[13]	87. Misleading Estimation About Skills Of Workers	*		[11]
24. Lack Of User Involvement			[13]	88. Lack Of Technical Feedback	*	*	[11]
25. Lack Of User Acceptance			[13]	89. Compromise On Profit To Save Name			[11]
26. High User Training Needs			[13]	90. Risk Of Economy Distortion		*	[11]
27. Large Project Size	*	*	[13]	91. Expansion Of Software Requirements	*	*	[15]
28. Hardware Constraints			[13]	92. Inaccurate Estimation Of Software Effort			[15]
29. Lack Of Reusable Components			[13]	93. Low Knowledge And Understanding Of Clients Regarding The Requirements	*	*	[9]
30. Lack Of Cost Controls	*	*	[13]	94. Incorrect Requirements			[9]
31. Lack Of Delivery			[1	95. Lack Of Frozen			[9]

Commitment			3]	Requirements			
32. Lack Of Requirements Stability			[13]	96. Undefined Project Success Criteria			[9]
33. Requirements NOT Complete And Clear			[13]	97. Conflicting System Requirements			[9]
34. Lack Of Testability		*	[13]	98. Conflict Between User Departments			[9]
35. Implementation Difficulty			[13]	99. Low Number Of Users In And Outside The Organization			[9]
36. High System Dependencies	*		[13]	100. Instability Of The Client's Business Environment			[9]
37. Lack Of Response Or Other Performance Factors			[13]	101. Dependency On A Few Key People			[9]
38. High Customer Service Impact			[13]	102. Lack Of Staff Commitment, Low Morale		*	[9]
39. Data Migration Required			[13]	103. Instability And Lack Of Continuity In Project Staffing			[9]
40. Lack Of Pilot Approach			[13]	104. High Number Of People On Team			-
41. Lack Of Alternatives Analysis			[13]	105. Low Team Diversity		*	[9]
42. Lack Of Quality Assurance Approach			[13]	106. Lack Of Organizational Maturity			[9]
43. 15Lack Of Development Documentation			[13]	107. Lack of Project leader's experience		*	[9]
44. No Use Of Defined Engineering Process			[13]	108. High Extent Of Changes In The Project	*	*	[9]
45. Late Identification Of Defects			[13]	109. Excessive Schedule Pressure			[9]
46. Bad Defect Tracking			[13]	110. Inadequate Cost Estimating		*	[9]
47. Lack Of Or Bad Change Control For Work Products			[13]	111. Poor Project Planning	*	*	[9]
48. Problem With Physical Facilities			[13]	112. Ineffective Communication			[9]
49. Problem With Hardware Platform	*		[13]	113. Improper Definition Of Roles And Responsibilities			[9]
50. Tools Unavailability			[13]	114. Need To Integrate With Other Systems			[9]
51. Bad Project Management Approach / Method	*	*	[13]	115. Inadequate Configuration Control			[9]
52. Lack Of Project Management Experience	*	*	[13]	116. Low Quality Of Software And Hardware Supplier Support			[9]
53. Bad Project Management Attitude			[13]	117. Excessive Reliance On A Single Development Environment			[9]
54. Lack Of Project Management Authority			[13]	118. High Extent Of Linkage To Other Organizations			-
55. Team Member Unavailability	*	*	[13]	119. Resource Insufficiency			[9]
56. Bad Or Low Mix Of Team Skills			[13]	120. Intensity Of Conflicts			[9]
57. Lack Of Experience With Software Engineering Process	*	*	[13]	121. Lack Of Control Over Consultants, Vendors ,sub-contractors			[9]

58. Lack Of Training Of Team	*	*	[13]	122. Massive User Stress			[10]
59. Lack Of Expertise With Application Area (Domain)	*		[13]	123. Lack Of Project Delivery Milestones			[10]
60. Development Technology NOT Match To Project			[13]	124. Over-optimistic Technology Perceives			[10]
61. Lack Of Development Technology Experience Of Project Team	*	*	[13]	125. Staff Turnover			[10]
62. Immaturity Of Development Technology			[13]	126. Backup Issues			[10]
63. High Design Complexity	*	*	[13]	127. Bad Preservation Of Intellectuals			[10]
64. Lack Of Support Personnel			[13]	128. Inability To Secure Confidential Customer Data			-