An Early Phase Software Project Risk Assessment Support Method for Emergent Software Organizations

Article *in* International Journal of Advanced Computer Science and Applications · May 2017

DOI: 10.14569/IJACSA.2017.080514

Sahand Vahidnia
Ankara University
2 PUBLICATIONS **0** CITATIONS
SEE PROFILE

Iman Askerzade
Ankara U.-Azerbaijan IofPhys
155 PUBLICATIONS **529** CITATIONS
SEE PROFILE

An Early Phase Software Project Risk Assessment Support Method for Emergent Software Organizations

Sahand Vahidnia
Computer Engineering Department
Ankara University
Ankara, Turkey

Ömer Özgür Tanrıöver
Computer Engineering Department
Ankara University
Ankara, Turkey

I.N. Askerzade
Computer Engineering Department
Ankara University
Ankara, Turkey

***Abstract*—Risk identification and assessment are amongst critical activities in software project management. However, identifying and assessing risks and uncertainties is a challenging process especially for emergent software organizations that lack resources. The research aims to introduce a method and a prototype tool to assist software development practitioners and teams with risk assessment processes. We have identified and put forward software project related risks from the literature. Then by conducting a survey to software practitioners of small organizations, we collected probability and impact of each risk factor opinions of 86 practitioners based on past projects. We developed a risk assessment method and a prototype tool initially based on data that accumulates further data as the tool. Along with a risk prioritisation and risk matrix, the method utilises fuzzy logic to provide the practitioners with predicted scores for potential failure types and aggregated risk score for the project. In order to validate the usability of the method and the tool, we have conducted a case study for the project risk assessment in a small software organization. The introduced method is partially successful at prediction of risks and estimating the probability of predefined failure modes.**

Keywords—Software Risk Identification; Software Risk Assessment; Failure Mode Prediction; Fuzzy Decision Support

I. Introduction

According to reports [1], the global software market is estimated to have a value of US$333 billion in 2016 which is expected to grow by 7.2%. On the other hand, the success rate of global (mainly US and Europe) software projects in 2015 is only 29% [2]. Therefore, it is highly desired to follow software engineering practices to prevent further loss in software spending. Among software development and engineering activities, risks assessment of software projects is a significant task, requiring effort and time. In many organizations, especially in small organizations, project managers do not have enough expertise and time for risk assessment. However, the consequences of ignoring this activity will result in loss of time and resources for the organization, as without risk assessment incorrect decisions can be made.

Although there are slight variations in definition of terms in the literature, a risk factor is a potential problem that may occur. Similarly in the software domain, risk is considered as an uncertain event or condition with negative or positive consequences on a software project on one or more project objectives such as scope, schedule, cost, and quality PMBOK [3]. It should be identified, assessed by its probability of occurrence and impact as its two important dimensions, and a contingency plan should be developed for remediating the problem when it actually occurs [4].

In accordance with above definition, various studies have been conducted and risk factors, categories [5], [6] and analysis tools [7] have been introduced. However, most developed methods and tools either cover a limited set of risk factor that potentially occur later in software project lifecycle or only focus on the improvement of a method/technique within the risk assessment process, such as aggregation, root cause analysis, etc. [8]–[10]. Most of the methods assume that the organization/team already has accurate near precise information about the project in the initial planning phase. Experience in risk identification, existence of a potential risk register and historical data is widely assumed. There are other studies focusing on a specific set of risk factors (Appendix 1), such as operational risk, requirements risk, etc. Furthermore, risk factors used in different studies may be disjoint or sometimes overlapping. In real world, software practitioners cannot benefit from these methods unless in a consolidate framework is provided. As for the available software tools, they are mostly enterprise, expensive and the rest only have limited predefined set of risks or no predefined risks at all. Furthermore, they do not provide any baseline and prediction on which the practitioner can use initially, benchmark his project, and predict potential failure types. Hence, there is a need to provide a consolidated method for the software practitioners of small organizations with scarce experience and resources. This will help them not to miss potential project risks, especially during early phases of the project. Related work section provides a more thorough review of the problems stated.

Therefore, one of our goals in this study is to put out a risk assessment method especially for practitioners of emergent software organizations with relatively low previous experience and historical data. To do this, from the literature, a wide coverage list of software project related risks was identified, which possibly rated at initial phase of the project with relatively little information. By conducting a survey to software practitioners, risk data have been collected; both in terms of impact and probability based on software practitioners' previous project experiences. In addition to risk prioritisation, the method assist the practitioners with an initial set of possible risks, probability and impact values to be revised for their specific project settings. Furthermore, based on the data provided by the risk assessor, the tool predicts probability of failure types, such as defects, overtime and over budget to the risk assessor. In order to validate the usability of the method and the tool, a case study was conducted for a project risk assessment in a small software organization. The conduct of this research is shown in Figure 1.

In the rest of the paper, first previous studies related to software project risk assessment is discussed. Then, the risk factors collection, the assessment method and the tool developed is described. Finally, the findings of the case study conducted for validations of the method and applicability of the tool is presented.

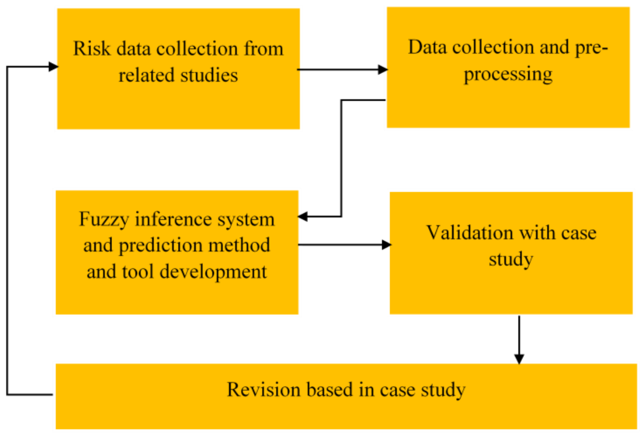

Risk data collection from related studies
Data collection and pre-processing
Fuzzy inference system and prediction method and tool development
Validation with case study
Revision based in case study

Fig. 1. Study flow

II. Related Works

During last three decades, various software risk factors, assessment and analysis tools have been introduced and explored. Majority of these studies can fit into three categories of: (1) Researches focusing on software risks and risk factors identification, (2) Researches focusing on aggregation methods of risk factors ratings, and (3) Researches proposing risk assessment and risk management tools and methods. In the following, some influential researches, specifically related to software project risk assessment are provided.

In an early study published by Software Engineering Institute [11], a risk management model and a taxonomy based software risk identification has been performed. The method consists of a taxonomy based questionnaire and a process for its application. The taxonomy provided a structure for organizing software development risks i.e. Product Engineering, Development Environment, and Program Constraints. Carr et al. demonstrated the application of risk management model in five repeating activities of identify, analyse, plan, track, control, and communicate at the centre of activities. It has been observed that adopting Pareto 20-80 rule is important and dealing with very first 20% of risks will be most effective highlighting the importance of risk prioritisation in a risk assessment method hence also considered in our method. This generally accepted approach is adopted in widely used text books in software project management [4].

Identification of software risks has been tackled in Li Xiaosong's study [5], providing a wide range of risk factors and categories in addition to general risk matrix and risk levels with their definitions. Another similar study by [12] contains a set of proposed risks in development phases with their definitions. Finally, Verner [6] has performed a literature review of available studies. The study has extracted 77 risk mitigation advises alongside with 85 risks. Using this and some other works, we aggregated sometimes disjoint or overlapping risk factors.

Due to inherent problem of difficulty in assigning numeric values to risk factors, as an example Markowski's study [9] proposed to fuzzify risks ratings. Risk ratings and values are fuzzified and fuzzy inference system (FIS) is adopted for processing and prediction. The problem of aggregation (linear aggregation has been found misleading) of risk scores has also been tackled. Choquet integral based aggregation approach to software development risk assessment [10] is an example of second category of related work. The study provides a software risk aggregation method to estimate the risk of a project. In addition to aggregation method, a set of risk factors, categories, and their associations have been developed. The study proposes a multi Choquet integral based multi attribute aggregation method for decision making process. For the same aggregation problem, [13] defined a method based on fuzzy logic. The goal of this study is early assessment of operational risks in software development. According to the study, before and during developing software, there are not enough data to have a full-scale risk assessment. So a fuzzy method is implemented to address the issue of uncertainty. In addition, a causal model is developed using fuzzy rules.

Among researches proposing software risk assessment methods, [14] proposes a method to statistically analyse and evaluate risk factors and their prices. The method enables to approximate risk-pricing parameters for four risk factors, namely, (1) Application Task, (2) Personnel Capability, (3) Process Maturity, and (4) Technology Platform. Hence, this study focuses on pricing dimension and in this respect granularity; hence the number of risk factors is small. In order to have a better software risk control, Hu's research [15] suggests planning based on causality. The proposed method is based on Bayesian Networks with Causality Constraints hence taking a probabilistic approach rather than fuzzy logic. In this study, in order to gather necessary data, a survey has been conducted which is similar to our approach. The data is used for constructing the Bayesian Networks. However, the paper mostly focuses on finding relations between variables rather than assessment. Bayesian Networks is used in numerous works due to their simplicity and ease of implementation [16]. But a solid amount of data is required for constructing a proper network.

Reference [17] proposed a risk assessment technique for evaluating risk levels in software projects through analogies with economic concepts. This study defines project risk levels as the probability of project's failure in achieving goals and evaluates risk levels using a risk identification questionnaire. Structurally this study is similar to ours in terms of comparing risks and effects, but the definition of risks and methods to handle them and comparing them is different. Costa et al. uses weighted normalised medians for risk factor, in contrast, we used FIS to compare risks and weights gathered in initial questionnaire. Finally, Costa et al. addresses the issue of financial loss and gain prediction, whereas our method uses project failure modes as predictions.

The relationships between project setting, governance and project success are studied recently in [18]. A survey has been conducted in attempt to prove the positive relationship between project management methodology (PMM) and project success. The relationship between the project management methodology and project success is moderated by project governance. The first hypothesis is shown to be valid showing the importance of the general project setting related risks for project success can be considered as valid hypotheses.

Recently, a similar study [19] implemented fuzzy method in aggregation of software risk factors. However, the application of the risk assessment is extraneous and differs in the method of data-point accumulations. In contrast to our study, this study relies only on 7 experts' data of the field and considers only a handful of risk factors. As a result, the study does not provide predictions according of expert data.

Further, tools potentially that can be used for software project risk assessment are publicly available and accessible. As an example, RisCal [7] is a proposed tool by Haisjackl. RisCal implements risk identification, risk analysis and risk prioritisation. In risk identification step, it allows for a user defined risk models in addition to the pre-defined risk models. There are also studies and tools with different approaches like esrcTool [20]. esrcTool implements FPA (Function Point Analysis) to estimate software cost and risks. The study focuses on functional breakdown of software rather than considering overall project environment and attributes. Hence, the method and tool focuses on more software product risks rather than project environment.

In the literature, each study adopts a set of risk factors for the study in question, sometimes completely disjoint or overlapping. Initially, most of the reviewed studies only consider a limited set of risks (not in terms of number, but also in terms of coverage over different aspects of software engineering processes), so a wider coverage of software project related risks is put forward which can potentially be assessed at initial phase of a project. Secondly, aggregation-focused studies are generally difficult to implement, as they require technical expertise to apply. Thirdly, available software tools mostly enterprise solutions only to manage a limited predefined set of risks or no predefined risks at all. Therefore, a risk assessment method and a prototype tool were developed so that they can be used by practitioners of small organizations with relatively low previous experience and historical data at an early phase of the project. The tool suggests the practitioners with a set of possible risks and their risk values for a specific project setting provides risk prioritisation with risk matrix, project risk level using fuzzy aggregation and potential failure type score using fuzzy inference. In this respect, our approach consolidates and supports the early risk assessment task at initial phase of a software project.

III. The Method and Pro-Type Tool

A. Risk Factors, Scales and Data

In various prior studies, risk factors considered focus on a specific aspect or phase of software development. We aimed at a risk factor set that can be used as a project initiation phase. So as the first phase, risk factors and categories were extracted from related studies [5], [12], [14], [21]–[25]. Therefore, a superset of 128 risks with a greater coverage was created. Then, similar and overlapping risks are unified. Furthermore, risk statements were changed into negative statements to ease practitioners understanding and ratings.

For the scales two components: probability and impact [7] is considered. Risk score is usually defined as the product of probability and impact [26]. Hence, scale definitions of probability and impact levels are reused from [21] as shown in Tables 1 and 2.

In addition, as an assessment tool a 3x3-risk matrix is used. Probability and impact are two dimensions of a risk matrix. As one of widespread tool for risk evaluation, risk matrix is natural to understand by evaluators. There are also other configurations of risk matrices like 5x5, 7x5 and 7x4 risk matrices which are not adapted due to less accurate information at early project phase and simplicity of 3x3 matrices [9]. Risk matrix dimensions or axes are divided into three level each, which creates a nine cell qualitative matrix [27]. This matrix has three parts: (Figure 2).

1) High/Major Concern (red): Risk is high in these sections and an action should be taken.

2) Medium/Concern (yellow): Risk is moderate in these sections and there is a chance that risks in these areas may affect project.

3) Low/No Concern (green): Risk in these sections are low and acceptable and can be ignored.

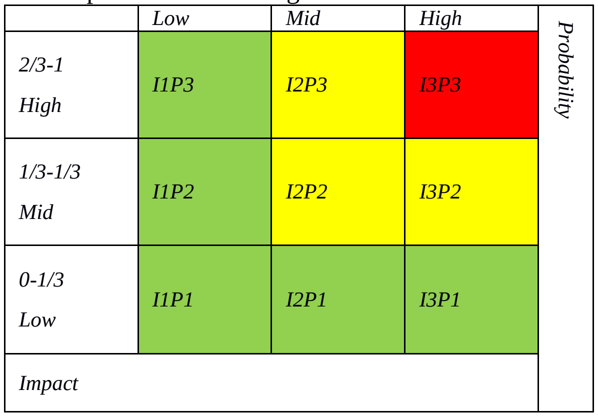


Fig. 2. Risk matrix and regions

TABLE I. Probability Level Definitions

Probability Levels	
High / Very Likely	High chance of this risk occurring, thus becoming a problem (x>%70)
Medium / Probable	Risk like this may turn into a problem once in a while (%30<x%70)
Low / Improbable	Not much chance this will become a problem (x<%70)

TABLE II. IMPACT LEVEL DEFINITIONS

Impact Levels	
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
Medium / Critical	Considerable problem with project with recoverable operational capacity; cost overrun exceeding 10% (but less than 50% of planned cost
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 time frame

Later, as a data gathering method a questionnaire was designed within the tool for surveying developers accessible online (*http://46.197.200.167/public_result.php*). It comprises three parts such that; first part obtains general information regarding a previous project considered by the practitioner such as type, size (approx. LOC), methodology used, etc. Second part contains failure and challenges of final project which contains 10 questions (Appendix B) These questions were adapted from previous studies [5], [13], [14], [21]–[25], [28], [29]. The last and 3rd part of the gathers information about the risk factor ratings for 128 risks. Based on this, initially, a risk matrix is generated using 86 practitioners' ratings. The most recent version of this matrix is publicly available at the web address mentioned earlier.

In contrast to a related study conducted previously [30], wider cross-correlations are analysed between risks. According to Weinberg [31], Pearson correlation coefficients of $r = \pm 0.5$ are considered strong and correlation coefficients close to ± 1 are the strongest. Evans recommends a correlation coefficient of ± 0.6 *to* ± 0.79 is considered a strong correlation [32]. As a result, to keep a safety margin, correlation coefficients which are among ± 0.6 *and* ± 1 are considered as strong. Table 3 demonstrates the highly cross correlated risk factors. Cross correlation creates a duplicate variable effect, which is not desired in the learning tools. Pearson correlation coefficients are obtained using Matlab software's [33] Pearson's correlation function of "*corrcoef()".* The Pearson correlation coefficient of two variables is measured as following where μ_x and σ_x are the mean and standard deviation of X:

$$\rho(A,B) = \frac{1}{N-1}\sum_{i=1}^{N}\left(\frac{\overline{A_i-\mu_A}}{\sigma_A}\right)\left(\frac{B_i-\mu_B}{\sigma_B}\right) \tag{1}$$

There are 48 highly correlated risk factor pairs, unique risk factors at left side. Statistically these 48 risks represent repeated data among 128 risk. These 48 risk factors may be eliminated from risk factor list. However, due to lack of enough data points for further analysis, it was decided to keep the 128 risk factors within the tool for now. When the definitions of risk factors were analysed, it was noticed that the most of high correlated risk factors do not have logical bounds - at least as far as we could observe, as correlation does not necessarily result in causation.

B. *Description of First Phase of the Method*

A multi-purpose method and tool is designed and implemented. The tool gathers information from experts and practitioners and produces a general risk matrix. It also can produce specific risk matrices for projects with varieties of project specifications. It calculates the overall project risk based on fuzzy aggregation and produces probabilities of 10 different failure types for the project based on fuzzy inference. The tool is developed using PHP scripts as a web based software to provide an easy and wide access. Figure 3 outlines functionality of the tool. Data from previous practitioners using the tool is gathered and pre-processed. This data may be referred as expert data later. The pre-processing includes filtering missing and inconsistent data. A general risk matrix is extracted from this data. Then practitioner input is taken for the project under assessment. Both data sets will go through Phase 1 and where initial risk matrix for practitioner is proposed. Then practitioner is allowed to alter proposed risks to get a more accurate risk matrix. Remark that, in case of initial projects risk assessment, it is difficult to measure risk quantitatively. As proposed by Xu [13], when dealing with qualitative variable (like low, mid, high), it is advised to work with fuzzy numbers. The altered and more accurate risk set will pass through Phase 2 for a failure mode analysis of the project.

In order to generate a risk matrix for practitioner, a module is designed to accumulate necessary data for risk matrix. In Phase 1 a query of data-points with parameters of Part I of survey is done. These parameters are "project size (LoC)", "project methodology", "project paradigm" and "development type". The result of is a filtered result of available practitioner data-points in form of a risk matrix and prioritisation. This filtered result come in form of averages of probabilities and impacts of selected data-points for all risks based on the prior parameters. Thus, a 3x3 risk matrix is generated from this data.

TABLE III. HIGHLY CORRELATED RISKS

High						**Very High**		
Risk ID 1	***Risk ID 2***	***Correlation Coefficient***	***Risk ID 1***	***Risk ID 2***	***Correlation Coefficient***	***Risk ID 1***	***Risk ID 2***	***Correlation Coefficient***
23	106	*0.6226*	9	54	*0.7072*	90	109	*0.8008*
41	108	*0.6242*	47	126	*0.7101*	101	111	*0.8029*
5	17	*0.6279*	44	95	*0.71029*	21	30	*0.8062*
92	123	*0.6338*	27	48	*0.71146*	37	51	*0.82162*
63	76	*0.63441*	26	11	*0.71208*	39	49	*0.82401*
35	48	*0.63496*	43	104	*0.71679*	18	125	*0.83277*
32	75	*0.65482*	28	95	*0.72975*	55	111	*0.84347*
12	66	*0.65666*	3	126	*0.73161*	25	74	*0.84403*
60	96	*0.66049*	36	126	*0.74008*	38	126	*0.84679*
52	103	*0.66168*	57	88	*0.74041*	24	50	*0.85583*
86	120	*0.6732*	85	120	*0.74108*	87	93	*0.85617*
68	74	*0.6806*	70	74	*0.74817*			
19	33	*0.68681*	97	115	*0.75964*			
79	73	*0.69163*	122	115	*0.76228*			
64	67	*0.69514*	65	120	*0.76273*			
58	117	*0.70219*	29	31	*0.77262*			
84	98	*0.70398*	71	93	*0.77567*			
105	124	*0.70634*	56	74	*0.78317*			
34	109	*0.70646*						

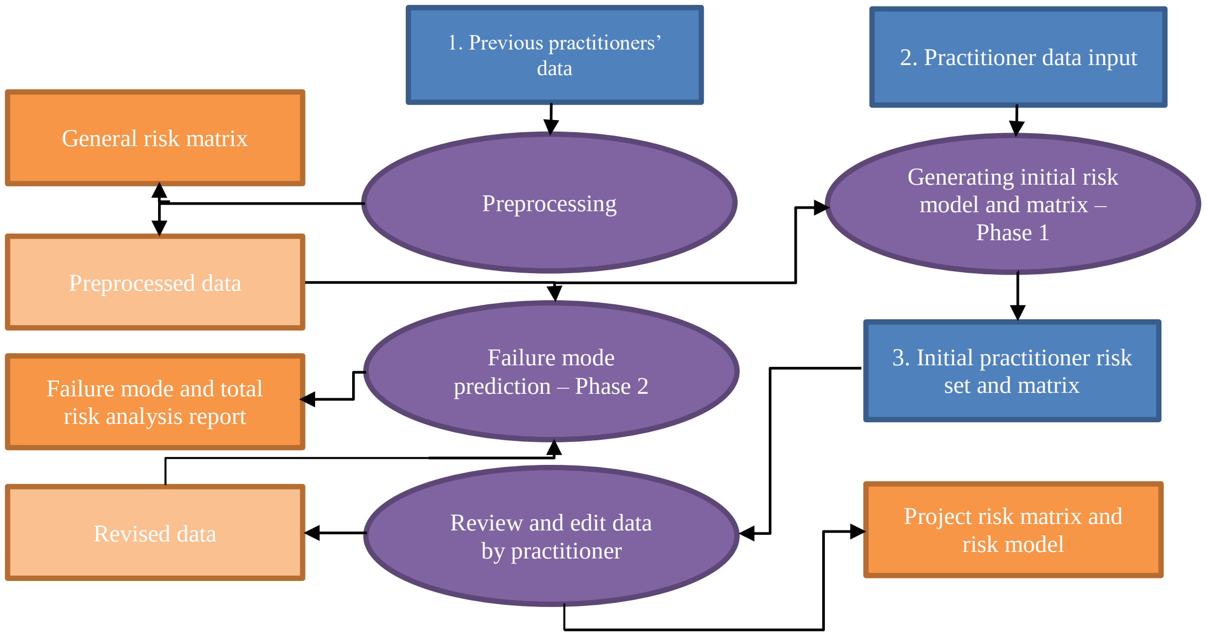

1. Previous practitioners' data
2. Practitioner data input
General risk matrix
Preprocessing
Generating initial risk model and matrix – Phase 1
Preprocessed data
Failure mode prediction – Phase 2
3. Initial practitioner risk set and matrix
Failure mode and total risk analysis report
Revised data
Review and edit data by practitioner
Project risk matrix and risk model

Fig. 3. Description of the Method

Proposed risk matrix in this phase is valuable for practitioners and will give them an initial and brief view of possible risks and their importance in similar projects. But, this will not exactly match to the specific project setting to rely on before their data is collected. However, the matrix and initial risk sorting will draw a helpful guideline for practitioner. Practitioner can change risk impact and probability values manually in order to achieve a better rating in next phase.

C. Description of the Second Phase of the Method

In the second phase, as proposed by Xu [13], when dealing with qualitative variable (like low, mid, high), it is advised to adopt fuzzy numbers. Second phase implements fuzzy logic to assess the risks and predict failure modes. A decision matrix is used to evaluate and rank the overall and partial failure score of the project, using practitioner's inputs or predicted risk scores in first phase based on previous data. Practitioner input is acquired in form of probabilities and impacts. Probability and impact scores are turned into triangular fuzzy numbers and aggregated.

Then Mamdani's inference model [9], [34] is used for prediction of failure types. To analyse failure modes, data points with negative scores for the failure mode are selected. For instance, in order to perform this selection, only the risks with a particular failure mode score of 3 out of 3 are taken as a match and remaining results are dismissed. In contrast, analysing overall project risk requires all data points.

Due to missing and imprecise information at initial phase of the projects, fuzzy decision matrix is used with triangular fuzzy numbers (TFN) [35]. Fuzzy decision matrix has less complexity and is effective for ranking fuzzy numbers. For membership function $\mu_i(x)$ of fuzzy number, $\tilde{n}_i$ can be defined as:

$$e_{ij} = max_{x \geq y}\left\{\min\left(\mu_i(x), \mu_j(y)\right)\right\} \tag{2}$$

$$for\ all\ i,j = 1,2,\dots,m$$

$\tilde{n}_i > \tilde{n}_j$ if and only if $e_{ij} = 1$ and $e_{ji} < Q$, where, Q is some fixed positive fraction less than 1.

First part of this equation requires expert data in form of fuzzy sets. To provide this, filtered data-points are sorted into two categories of probability and impact, with each containing risk factor scores. The risk factor scores go through fuzzification process (*see* Equation (3) and Figure 4) by a membership function for each corresponding risk factor.

$$\mu_A(x) = \begin{cases} 0 & if\ x < a \\ \frac{x-a}{b-a} & if\ \ a < x \leq b \\ \frac{c-x}{c-b} & if\ \ b < x \leq c \\ 0 & if\ x > c \end{cases} \tag{3}$$

$$A = (a, b, c)$$

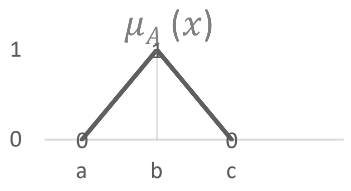


Fig. 4. Fuzzy membership function illustration

After fuzzification of probability and impact, scores in the data-points are aggregated to form a single expert opinion data. To do so, an aggregation operator is adopted from Pandey [36] which is based on arithmetic means of L-Apex and R-Apex Angles of TFN.

$$\bar{b} = \frac{1}{n}\sum_1^n b_i \tag{4}$$

$$\bar{a} = \frac{1}{n}\sum_1^n b_i - \tan\left[\frac{1}{n}\sum_1^n tan^{-1}\left(b_i - a_i\right)\right] \tag{5}$$

$$\bar{c} = \frac{1}{n}\sum_1^n b_i + \tan\left[\frac{1}{n}\sum_1^n tan^{-1}(c_i - b_i)\right] \tag{6}$$

Risk value is obtained by calculating the product of probability and impact values; the method of Shang [37] with adjustments is used to calculate the multiplication of triangular fuzzy probability and impact numbers. Remark that there are other works including Taleshian's method, [38] which uses trapezoidal numbers and could be also used with some adjustments. Hence, multiplication of $\mu_A(x)$ and $\mu_B(y)$ can be obtained using (7).

$$\mu_{\tilde{Q}}(Z) = \begin{cases} \dfrac{-(a_1b_2 + a_2b_1 - 2a_1a_2) + \sqrt{(a_1b_2 - a_2b_1)^2 + 4(b_1 - c_1)(b_2 - c_2)Z}}{2(b_1 - a_1)(b_2 - a_2)} \\ a_1a_2 \le Z \le b_1b_2 \\ \dfrac{-(c_1b_2 + c_2b_1 - 2c_1c_2) - \sqrt{(a_1b_2 - a_2b_1)^2 + 4(b_1 - c_1)(b_2 - c_2)Z}}{2(b_1 - c_1)(b_2 - c_2)} \\ b_1b_2 < Z \le C_1C_2 \\ 0 \qquad otherwise \end{cases} \tag{7}$$

For processing and fuzzification of data, membership functions must be defined. The fuzzy numbers must be triangular to match the Equation (3), so can be applied to aggregation and multiplication equations.

To prevent misinterpretation of results, probability and impact values are gathered in quantitative range of [0-8] with initial peak points in range of [2-6]. Otherwise, aggregation and multiplication equations could lead in to due to producing negative and non-triangular fuzzy number. Remark that normally, the probability is expected to be evaluated in [0-1] range. However, we use the probability score as a variable to be rated by the practitioner and transform it as a fuzzy number, therefore it may range between 0 and 8 during calculations in the method. Impact score are calculated in the same manner. Now multiplication (for measuring total risk) produces fuzzy numbers from 0 to 64 which can contain any triangular fuzzy numbers produced using introduced techniques. Figure 5 demonstrates our predefined fuzzy membership functions, which L stands for low, M for medium and H for high probability or impact:

L=(0.5,2,3.5) , M=(2.5,4,5.5) , H=(4.5,6,7.5)

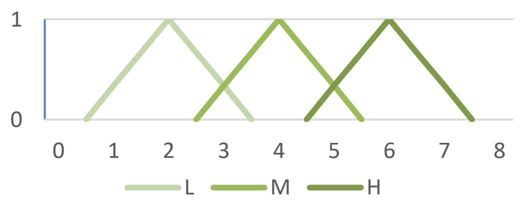


Fig. 5. Fuzzy membership function

D. An Example to Illustrate Second Phase of the Method

To clarify our method, a failure mode prediction example is given. We assume a risk set with corresponding scores of probability and impact described as below. Given the probability and impact scores of L (low) and M (mid) for "Lack of Development Technology Experience of Project Team" risk, fuzzy numbers for these values can be obtained using membership functions defined previously.

Probability L:

[0/0, 0.34/1, 1/2, 0.34/3, 0/4, 0/5, 0/6, 0/7, 0/8] (8)

Impact M:

[0/0, 0/1, 0/2, 0.34/3, 1/4, 0.34/5, 0/6, 0/7, 0/8] (9)

Using Equation (7) and defined membership functions, we can calculate combined fuzzy risk score for risk of "Lack of Development Technology Experience of Project Team" in this case. *See* Equation (10).

R = [0/0, 0/1, 0.15/2, 0.33333333/3, 0.49/4, 0.63/5, 0.76/6, 0.88/7, 1/8, 0.89/9, 0.78/10, 0.69/11, 0.59/12, 0.50/13, 0.41/14, 0.33/15, 0.25/16, 0.17/17, 0.09/18, 0.01/19, 0/20, 0/21, 0/22, 0/23, 0/24, 0/25, 0/26, 0/27, 0/28, 0/29, 0/30, 0/31, 0/32, 0/33, 0/34, 0/35, 0/36, 0/37, 0/38, 0/39, 0/40, 0/41, 0/42, 0/43, 0/44, 0/45, 0/46, 0/47, 0/48, 0/49, 0/50, 0/51, 0/52, 0/53, 0/54, 0/55, 0/56, 0/57, 0/58, 0/59, 0/60, 0/61, 0/62, 0/63, 0/64] (10)

After calculating the expert and practitioner values by equation (2), minimum values of each risk among test and expert data is obtained as a vector of fuzzy numbers. Later, maximum values of fuzzy risk scores are used to obtain a single aggregated fuzzy number. This number is the total failure score for provided test data. Higher score means the chance of failure is also higher. The same method applies to all failure modes, but it's important to remember that all risks must be considered and the risk of "Lack of Development Technology Experience of Project Team" has been given only to demonstrate how a single risk is being handled in the method. Likewise, this matrix can potentially point out the most influential risk factors. After processing e_{ij} at Equation (2), result is defuzzified. Defuzzification for risk index can be expressed by Equation (11).

$$a = \frac{\sum \mu_A(x_i)x_i}{\sum \mu_A(x_i)} \tag{11}$$

Defuzzification of (10) using Equation (11) will result in (12):

$$a = \frac{69.5348}{9.0199} = 7.709 \tag{12}$$

This way, the defuzzified and final risk score for this example is computed, which is *7.709*. As discussed earlier, this score is also in range of [0-64] as expected. This number may also be scaled to *12.04%* to make the result more natural to interpret by the practitioner. Higher values are representing higher risks scores and lower values are representing lower risk scores.

IV. Case Study

In order to put the applicability of the proposed tool and method, we have conducted a case study. In the case study, the extent of support and usefulness of the tool and provided predictions are meant to be explored. The tool does not include risk responses. Thus, it is not expected to do a complete risk management, but only a prediction and assessment in risks and failure modes. The case must be able to meet the target project specifications as explained in early sections.

As a case, we had to find a case project with small development team with relatively low resources and little experience in risk assessment. Additionally, assessment of a project with Agile methodology was desired, as most of the small organizations prefer agile approaches. Thirdly, the project must be in early steps of development so the practitioners have to guess the risk levels without measuring the actual risks and failures. Otherwise, the result can be biased and misleading. Data collection in this case study is conducted in a first degree, direct (interview) [39] manner. It would be preferable to perform this interview in second degree, but due to limitations explained in next sections, this interview was performed with interactions.

In the case study, we tried to answer the following planned questions. Answering these questions can help us explore the validity and quality of method and the tool. These questions are:

1) Does this tool provide expected risk list for emergent software organizations?

2) Are there any missing important risks?

3) Does the proposed risk model represent real (possible) risk levels according to prior estimations?

4) How difficult is it to use the tool?

5) How long does it take to make an initial assessment of risks and failure modes?

6) How realistic and accurate failure mode predictions are?

7) Does the tool provide necessary insight for emergent software organizations?

B. Setting of the Case

As our method is opting to assist emergent software organizations with relatively less experience and knowledge in Software Risk Assessment at initial phase of a project, after considering three organizations, a small software company in a University Technology Zone is agreed to participate in the case study. The characteristics of this organization matched with the definition of immature software organization given by Paulk [40], [41]. This organization has seven personnel primarily working as a subcontractor for a larger organization developing solutions for a government organization. Hence, the organization has relatively little experience (only 5) on independent software development projects. However, recently they obtained an independent contract for developing an Emergency Triage [42] Decision Support Software for the University Hospital.

The goal of the project is to develop triage decision support software. This software should be able to categorize patients after the “Triage Nurse” initially evaluates them when they arrive to the emergency department. A patient is categorized into a priority class based on a triage nurse’s inputs and based on medical checks. The triage system that the software will implement is an already proven and accepted method, namely, the Canadian Triage and Acuity System (CTAS) [42] 5-level systems, with 5 priority categories. The software is only meant to serve as assistance, it should never take control from the user, as he/she should be able to override the software actions through his/her own professional judgment. At any time, the systems results can be overridden and life critical patients will be intervened outside of the system scope. Furthermore, the system will be delivered as a prototype and will not be fully operational until complete validation and verification; fully operational system will be developed if accepted.

The system is planned and developed by using SCRUM [43] by a team of two developers and a team leader (SCRUM master). Intensive commitment exists from the part of the emergency department management and highly dedicated involvement during development is established by assigning two emergency experts for the development. A Java based framework is planned to be used in order to minimise portability problems. In order to facilitate user interface development, user interfaces are planned to be developed with Jigloo GUI Builder [44]. The triage system is planned to be integrated into the hospital’s information system should be able to acquire patient medical history to aid the triage process. As the database, MySQL is planned to be used. The reason for these choices is previous expertise on the technology of the team or ease of integration with the hospital information system.

As there is not a formally defined risk management process in SCRUM the team has not conducted a traditional risk assessment. However, they have defined an initial set of 15 use cases as high level requirements such as View non-triaged patients, View triaged patients, Triage a patient, View patient medical history, etc. They have agreed that 3 (such as calculate triage category and assign treatment order of patient) of the 15 requirements will be more difficult to develop. They have foreseen to conduct state based verification for the critical objects within the scope of these requirements. However, they do not have any risk assessment output for the general software project risks. This is more or less typical for small teams working for with an agile methodology. They have considered a set of tools from the search engine including Jira [45], Risk Radar [46] and Risk Management Studio [47]. Most of these risk management tools will not provide a predefined set of risk and probable results, except for a number of risks limited to area like security. In contrast, our tool provides initial predefined software project risk factors with probability and impact levels based project attributes.

C. Conduct of the Risk Assessment with the Tool

The risk assessment is conducted with the team lead and developers and the method/tool developer in a 3-hour meeting. As the tool is currently in prototype state, the tool developer was present in order to explain the details of the use tool and explanation of the terminology used in the proposed risk register and use of the decision support techniques implemented. A set of informally prepared documents related to the scope of requirements of the project and system proposal for the bid were present. Each risk item is evaluated for its probability and impact level by the team and agreed with various discussions. As most of the information in the discussion was tacit, the referral to documents was little.

The tool’s graphical user interface has two stages. The first stage acquires general project information. In Figure 6, we provided the tool with this information and saved the progress.

Later input stage of the tool is the evaluation stage with default probability and impact levels and practitioner defined probability and impact levels as demonstrated in Figure 7. These scales are continuous that are called “visual analogue scales” [48] which give the practitioner better control and comfort in ratings. After customizing probabilities and impact scales according to the case, the tool generates a risk matrix for new data alongside the risk matrix for default data.

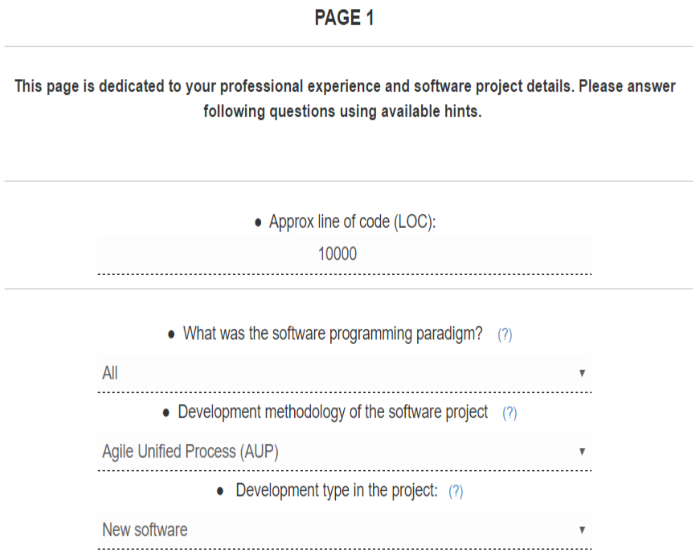
The team lead has made following observations during the conduct of the assessment:

1) Initial risk register (128 risk) embedded in the method/tool was useful for them different from any tools the team lead had used in his previous experience such as Jira, Radar, etc. He agreed that most of the risk factors might have impact on the software projects in general.

2) The historical data gathered from other practitioners used for generating approximate probability and impact levels helped the team to elaborate on their rating decision of impact and probability levels for each risk. In addition, the initial risk register provided guidance to rate risk factors for which they were unable to give levels either due to missing information or lack of consensus.

3) Risk prioritisation automatically generated by the tool will help them to focus risk remedial actions in a more focused and efficient way.

4) As scale rating, provided by a slider as 0-10 implicit levels was easy to assign for the practitioner intuitively rather than giving discrete 0-3 ratings. This visual assignment with more adjustments to the initial historical ratings eased the assign changes. (Note: This also provides us to make better predictions for specific failure types by the fuzzy prediction algorithm.)


PAGE 1

This page is dedicated to your professional experience and software project details. Please answer following questions using available hints.

- Approx line of code (LOC): 10000
- What was the software programming paradigm? (?) All
- Development methodology of the software project (?) Agile Unified Process (AUP)
- Development type in the project: (?) New software

Fig. 6. Data register stage screenshot

5) In fact, historically generated risk ratings were proposed by the tool to assist decision making, it became clear that for specific project and organization setting, there could be radical differences for some of the risks. Figure 7 demonstrates these differences in this case.

6) The team agreed that some of the risks they have not really thought about the triage project risk existed such as Backup Issues, Potential Increase in database size, and Security Risks.

7) Overall risk score of the project calculated by the tool and potential failure type estimation provided by the tool may be used as adjustment factor for project cost, schedule and resource.

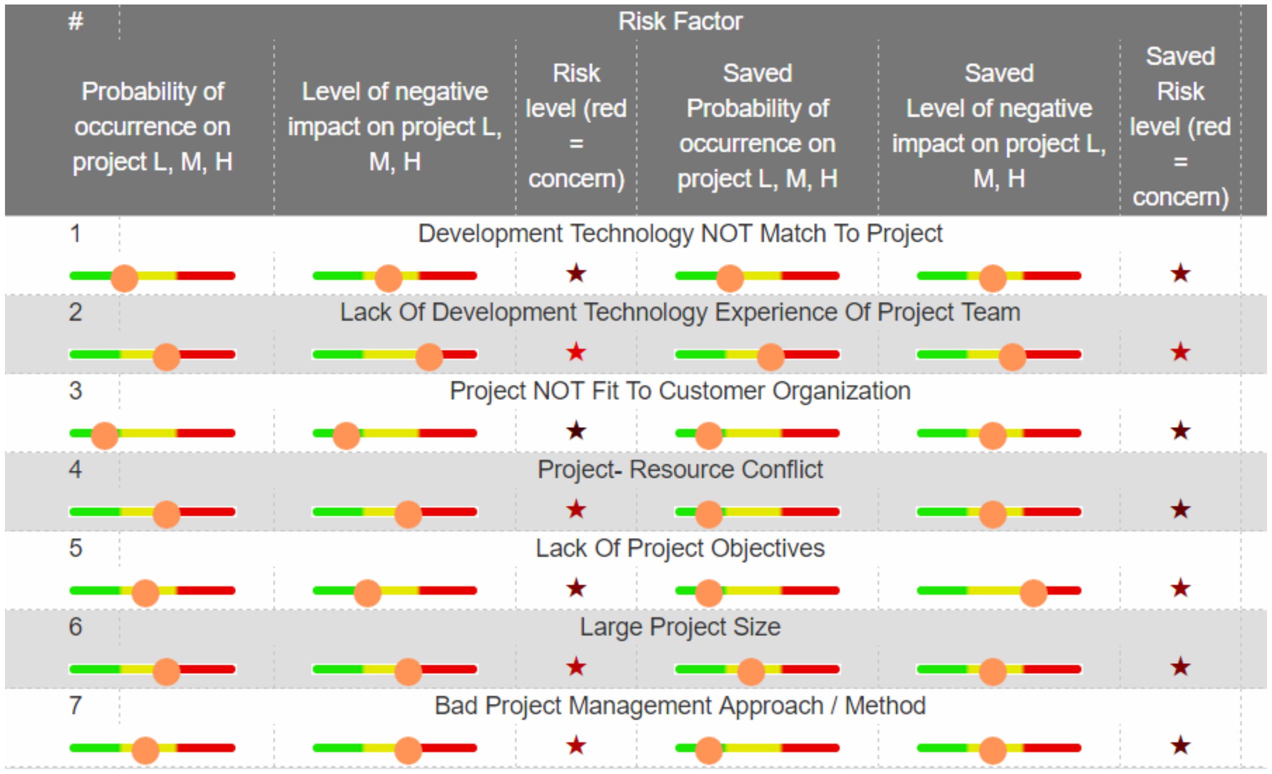

#
Risk Factor
Probability of occurrence on project L, M, H
Level of negative impact on project L, M, H
Risk level (red = concern)
Saved Probability of occurrence on project L, M, H
Saved Level of negative impact on project L, M, H
Saved Risk level (red = concern)
1 Development Technology NOT Match To Project
2 Lack Of Development Technology Experience Of Project Team
3 Project NOT Fit To Customer Organization
4 Project- Resource Conflict
5 Lack Of Project Objectives
6 Large Project Size
7 Bad Project Management Approach / Method

Fig. 7. Evaluation stage screenshot

D. Results

According to case data, High/Major Concern risks in custom risk matrix are available at Table 4. In addition, total quantitative risk score for case project is 50.30 that verbally can be categorized as "high" level. The qualitative value of high is a fuzzy value. A fuzzy value solves the problem of uncertainty with uncertain answer. For instance, Figure 9 shows a peak point of 50/64 and 51/64 with the lowest point of 0.1 in 33/64. It means the risk can be called 50/64 and 51/64 risky (nearly high) most of the times, and with a very low possibility it can be called 33/64 risky (near mid). It also means the risk cannot be categorized as sub-mid and below 32/64 at all. The same also applies to all other fuzzy numbers in a similar manner.

As mentioned in early sections, a failure mode analysis is also provided (Figure 8). According to the analysis of case data, the risk of failure for "Project Over-Schedule" in the case is low to medium. Result of this analysis is represented in Figure 9. The result of failure mode analysis in a single iteration of risk assessment may not provide necessary information regarding the possibility of failure, as these results are more like relative results than absolute.

This means the inference model is meant to be used as comparison model than an evaluating model. For a better comprehension regarding the project's failure mode probabilities, all failure modes are calculated -using the tool- and compared. The calculations are done using both proposed risk values and practitioner defined risk values of the case. Figure 10 is a demonstration of the comparison.

TABLE IV. MAJOR RISKS IN PROJECT

Risk	Region	Risk	Region
Low Knowledge and Understanding of Clients Regarding the Requirements	I3P3	Instability and Lack of Continuity in Project Staffing	3
Team Member Unavailability	I3P3	Lack of Expertise with Application Area (Domain)	3
Staff Turnover	I3P3	Dependency On a Few Key People	3
High Extend of Changes in The Project	I3P2	Lack of Organizational Maturity	3
Lack of Requirements Stability	I3P2	Need to Integrate with Other Systems	3
Lack of Frozen Requirements	I3P2	Excessive Reliance On a Single Development Environment	3
Requirements NOT Complete and Clear	I3P2	Misleading Estimation About Skills Of Workers	2
Expansion of Software Requirements	I3P2	Gold Plating	2
Lack of Software Developer Competence	I2P3		

As demonstrated in Figure 10, all failure mode ratings are in the range of 14/64 and 19/64 which can be represented in form of 22% - 29%. It can be concluded from the results that all failure modes are pretty far from being high, but relatively "Defects in Application" is more probable to occur than the rest. This can help the developer to generate a response to "Defects in Application" failure mode. This failure mode has been marked as relatively high in both predicted data and practitioner data. These results are proposed to guide the developing teams to take more precautions regarding the related risks. But for a better observation, it is recommended for developing teams to keep observing the risks and performing failure mode analysis in every step of the development. Failure mode values are mostly intended to be used as a comparison value of a failure mode in different time spans.

In a risk management cycle, it is very important to create responses for risks. As for this study, the response analysis is out of scope, but in this case study we decided to produce some suggestions to emulate a real risk management condition. Table 5 is a brief demonstration of possible and suggested responses in literature [6], without considering root causes. It is important to point out that only risk responses addressing the root cause of some, namely organizational risks may be truly effective [49]. However, this study does not provide a root cause analysis. Therefore, it is not expected to have an accurate risk response analysis.

TABLE V. RISK RESPONSES EXAMPLE

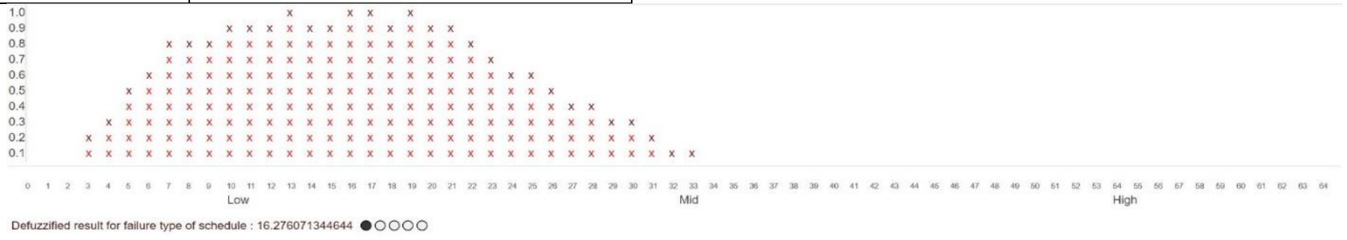
Risk	**Response**
Low Knowledge and Understanding of Clients Regarding the Requirements	Apply personal with domain knowledge. Define a person responsible for requirements specification and prioritization.
Lack of Software Developer Competence	Ensure that there is appropriate technical ability. Take into account the developers' skills assigning tasks.
Staff Turnover	At project start up, define undisputed areas of responsibility for all participants as well as the relational roles being instituted people management
Misleading Estimation About Skills Of Workers	The management should have a concrete description about the capabilities of each member of development team while estimating for the scope, size, and cost of the project avoiding optimistic estimations.

V. VALIDITY

There are threats to validity and we try to address them based on categories of validity threats which are pointed out by [39]. First threat to validity (Construct validity) of this case can be considered as possible misinterpretations of risks during the assessment. Subject practitioners might misinterpret the questions under normal circumstances, but in this case study an interview was conducted in first degree and interruptions were made during the interview to assure correct interpretations.

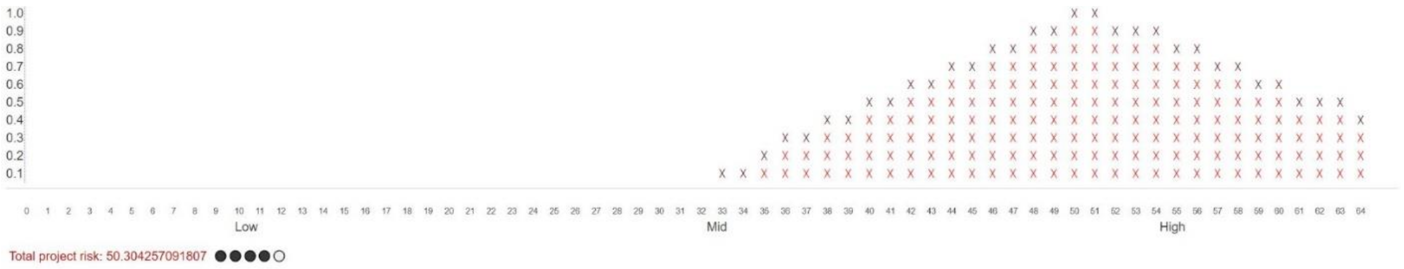
Another validity issue (internal) is correctly predicting risk factors and failure modes. No logical link is considered. The relations are indirectly established by data and via prediction method introduced. To get the more valid results and facilitate the use of the method, it is may be desirable to reduce risk factors as high cross correlations are observed. With further data, the method and relations can be improved. Also as pointed out by [50], it is not advised to use too many criteria in FIS. Thus, reduction of dimensionality in risk factors is expected to be effective in further validity of prediction.

The case study is only valid for projects with agile methodology and organizations with lower maturity levels (1 or 2) and cannot be generalised any further. Extending the project setting further can be a threat to the validity (External validity) of this case study. In order to extend the case study further, data must be improved to cover wider project settings. This research proposes risk assessment method and tool that the results might alter with different input data, but the logic behind the method will not change. It is important for future interested researchers to consider input data for training of the tool and do not rely on the exact same outputs. As mentioned earlier, this case study and whole study can be improved by improving input data and the validity of the tool improves as the data set improves.



Defuzzified result for failure type of schedule : 16.276071344644 ●○○○○

Fig. 8. Overschedule failure model risk



Total project risk: 50.304257091807 ●●●●○

Fig. 9. Total project risk

VI. Conclusion

In this study, we introduced a tool for small sized software development teams, with ability of providing initial risk set and rating recommendations. Additionally, we provided a fuzzy method based tool to facilitate the risk assessment by factors and their consequences in form of failure mode analysis. In addition, the method produces an overall project risk rating. All this information is useful for small-scale software companies with limited resources, especially at project bid, initiation phases and acceptance decisions.

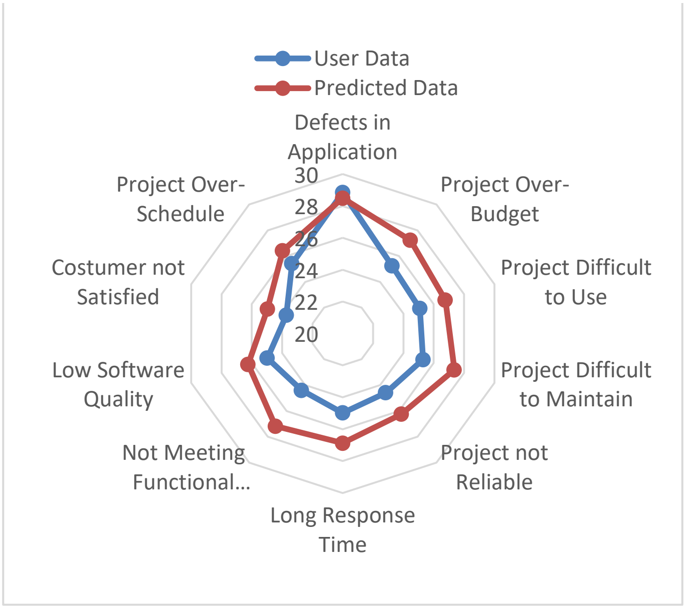


Fig. 10. Failure modes results

As explained in this case study, proposed risks and predicted scores are in mid to high level which are close to expert expectations. Another observation is based on comparisons of the automatically predicted failure mode scores (which are based on initial and automatically suggested risk ratings) and the predicted failure mode scores from practitioner manually altered input data shows a similar pattern in relatively high and relatively low failure mode scores. For instance, in both predicted failure mode scores (practitioner altered and automatically generated), the failure mode of "Defects in Application" poses a higher threat to the project and failure mode of "Customer not Satisfied" poses a lower threat to the project. Thus, in an overall conclusion, the method provides strong guidelines regarding the risk for practitioners and the steps of identifying, analysing and tracking risks. The method can possibly predict most common failure modes according to project data.

The tools risk rating proposal and prediction accuracy will certainly improve and results that are more generalisable may be drawn, as the usage of the tool by practitioners will increase the number of data points used by the tool. In addition, prediction method has potential for further improvements in order to point out influential risk factors for various failure modes. Additionally, a deeper study on risks and their characterisations can be conducted similar to [51] in order to have better risk control and management phases in future studies. It is also planned to provide root cause study and therefore a risk response advice in next version.

Acknowledgment

We want to thank practitioners involved in the survey and case study who donated their precious time to us. This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

References

[1] "Gartner Worldwide IT Spending Forecast," Gartner, Inc., 2016. [Online]. Available: http://www.gartner.com/newsroom/id/3482917.
[2] "CHAOS Report 2015," 2015.
[3] Project Management Institute, A guide to the project management body of knowledge (PMBOK ® guide). 2013.
[4] R. S. Pressman, Software Engineering A PRACTITIONER ' S APPROACH, 7th ed., vol. 33. 2010.
[5] L. Xiaosong, L. Shushi, C. Wenjun, and F. Songjiang, "The Application of Risk Matrix to Software Project Risk Management," 2009 Int. Forum Inf. Technol. Appl., pp. 480–483, May 2009.
[6] J. M. Verner, O. P. Brereton, B. a. Kitchenham, M. Turner, and M. Niazi, "Risks and risk mitigation in global software development: A tertiary study," Inf. Softw. Technol., vol. 56, no. 1, pp. 54–78, Jan. 2014.
[7] C. Haisjackl, M. Felderer, and R. Breu, "RisCal -- A Risk Estimation Tool for Software Engineering Purposes," 2013 39th Euromicro Conf. Softw. Eng. Adv. Appl., pp. 292–299, Sep. 2013.
[8] L. Yu and H. Liu, "Feature Selection for High-Dimensional Data: A Fast Correlation-Based Filter Solution," Int. Conf. Mach. Learn., pp. 1–8, 2003.
[9] A. S. Markowski and M. S. Mannan, "Fuzzy risk matrix," J. Hazard. Mater., vol. 159, no. 1, pp. 152–157, 2008.
[10] G. Büyüközkan and D. Ruan, "Choquet integral based aggregation approach to software development risk assessment," Inf. Sci. (Ny)., no. 180, pp. 441–451, 2010.
[11] M. Carr, S. Konda, I. Monarch, F. Ulrich, and C. Walker, "Taxonomy-based risk identification," Softw. Eng. Inst., no. June, pp. 1–24, 1993.
[12] H. Hizazi, N. H. Arshad, A. Mohamed, and Z. M. Nor, "Risk Factors in Software Development Phases," Eur. Sci. J., vol. 10, no. 3, pp. 213–232, 2014.
[13] Z. Xu, T. M. Khoshgoftaar, and E. B. Allen, "Application of fuzzy expert systems in assessing operational risk of software," Inf. Softw. Technol., vol. 45, no. 7, pp. 373–388, May 2003.
[14] A. Appari and M. Benaroch, "Monetary pricing of software development risks: A method and empirical illustration," J. Syst. Softw., vol. 83, no. 11, pp. 2098–2107, Nov. 2010.
[15] Y. Hu, X. Zhang, E. W. T. Ngai, R. Cai, and M. Liu, "Software project risk analysis using Bayesian networks with causality constraints," Decis. Support Syst., vol. 56, pp. 439–449, Dec. 2013.
[16] M. Perkusich, G. Soares, H. Almeida, and A. Perkusich, "A procedure to detect problems of processes in software development projects using Bayesian networks," Expert Syst. Appl., vol. 42, no. 1, pp. 437–450, 2015.
[17] H. R. Costa, M. de O. Barros, and G. H. Travassos, "Evaluating software project portfolio risks," J. Syst. Softw., vol. 80, no. 1, pp. 16–31, 2007.
[18] R. Joslin and R. Müller, "Relationships between a project management methodology and project success in different project governance contexts," Int. J. Proj. Manag., vol. 33, no. 6, pp. 1377–1392, 2015.
[19] C. Samantra, S. Datta, and S. S. Mahapatra, "Risk assessment in IT outsourcing using fuzzy decision-making approach: An Indian perspective," Expert Syst. Appl., vol. 41, no. 8, pp. 4010–4022, 2014.
[20] M. Sadiq, A. Rahman, S. Ahmad, M. Asim, and J. Ahmad, "EsrcTool: A tool to estimate the software risk and cost," in 2nd International Conference on Computer Research and Development, ICCRD 2010, 2010, pp. 886–890.

[21] S. Zardari, “Software Risk Management,” 2009 Int. Conf. Inf. Manag. Eng., pp. 375–379, 2009.

[22] B. Shahzad, I. Ullah, and N. Khan, “Software Risk Identification and Mitigation in Incremental Model,” 2009 Int. Conf. Inf. Multimed. Technol., pp. 366–370, 2009.

[23] D. Wu, H. Song, M. Li, C. Cai, and J. Li, “Modeling Risk Factors Dependence Using Copula Method for Assessing Software Schedule Risk,” in Software Engineering and Data Mining (SEDM), 2010 2nd International Conference on, 2010, pp. 571–574.

[24] H. Song, D. Wu, M. Li, C. Cai, and J. Li, “An entropy based approach for software risk assessment: A perspective of trustworthiness enhancement,” Softw. Eng. …, pp. 575–578, 2010.

[25] B. Shahzad and A. S. Al-Mudimigh, “Risk Identification, Mitigation and Avoidance Model for Handling Software Risk,” 2010 2nd Int. Conf. Comput. Intell. Commun. Syst. Networks, pp. 191–196, Jul. 2010.

[26] K. Olid and B. Mannan, “A Review of Software Risk Management for Selection of best Tools and Techniques,” pp. 773–778, 2008.

[27] G. Stoneburner, A. Goguen, and A. Feringa, “Risk Management Guide for Information Technology Systems : Recommendations of the National Institute of Standards and Technology,” Natl. Inst. Stand. Technol., no. 800–30, pp. 1–25, 2002.

[28] Ö. Hazir, “A review of analytical models, approaches and decision support tools in project monitoring and control,” Int. J. Proj. Manag., vol. 33, no. 4, pp. 808–815, 2015.

[29] A. A. Keshlaf and S. Riddle, “Risk Management for Web and Distributed Software Development Projects,” 2010 Fifth Int. Conf. Internet Monit. Prot., pp. 22–28, 2010.

[30] S. Vahidnia, Ö. Tanrıöver, and I. N. Askerzade, “AN EVALUATION STUDY OF GENERAL S OFTWARE P ROJECT RISK BASED ON SOFTWARE,” IJCSIT, vol. 8, no. 6, pp. 1–13, 2016.

[31] S. Weinberg and S. Abramowitz, “Statistics using SPSS: An integrative approach,” 2008.

[32] [32] J. D. Evans, Straightforward Statistics for the Behavioral Sciences. Brooks/Cole Publishing Company, 1996.

[33] The MathWorks Inc., “MATLAB.” The MathWorks Inc., 2015.

[34] E. H. Mamdani, “Application of fuzzy logic to approximate reasoning using linguistic synthesis,” IEEE Trans. Comput., vol. C-26, no. 12, pp. 1182–1191, 1977.

[35] E. Triantaphyllou and L. Chi-Tun, “Development and evaluation of five fuzzy multiattribute decision-making methods,” Int. J. Approx. Reason., vol. 14, no. 4, pp. 281–310, 1996.

[36] M. Pandey, N. Khare, and S. Shrivastava, “New Aggregation Operator for Triangular Fuzzy Numbers based on the Arithmetic Means of the L- and R-Apex Angles,” Submitt. Publ., vol. 2, no. 3, pp. 990–992, 2012.

[37] S. Gao, Z. Zhang, and C. Cao, “Multiplication operation on fuzzy numbers,” J. Softw., vol. 4, no. 4, pp. 331–338, 2009.

[38] A. Taleshian and S. Rezvani, “Multiplication Operation on Trapezoidal Fuzzy Numbers,” J. Phys. Sci., vol. 15, no. December, pp. 17–26, 2011.

[39] P. Runeson and M. Höst, “Guidelines for conducting and reporting case study research in software engineering,” Empir. Softw. Eng., vol. 14, no. 2, pp. 131–164, 2009.

[40] Ö. Tanrıöver and O. Demirörs, “A process capability based assessment model for software workforce in emergent software organizations,” Comput. Stand. Interfaces, vol. 37, pp. 29–40, 2015.

[41] M. C. Paulk, B. Curtis, M. B. Chrissis, and C. V. Weber, “The capability maturity model for software,” Softw. Eng. Proj. Manag., p. 48, 2006.

[42] [42] K. V. Iserson and J. C. Moskop, “Triage in Medicine, Part I: Concept, History, and Types,” Ann. Emerg. Med., vol. 49, no. 3, pp. 275–281, 2007.

[43] [43] K. S. or. Schwaber, “Scrum,” 2016. [Online]. Available: https://www.scrum.org/.

[44] Eclipse, “Jigloo SWT/Swing GUI Builder,” Eclipse Foundation, 2014. [Online]. Available: https://marketplace.eclipse.org/content/jigloo-swtswing-gui-builder. [Accessed: 06-Feb-2016].

[45] Atlassian, “JIRA Software,” Atlassian Foundation, 2016. [Online]. Available: https://www.atlassian.com/software/jira. [Accessed: 03-Feb-2016].

[46] Pro.Concepts, “Pro Concepts,” Pro Concepts LLC, 2014. [Online]. Available: http://www.proconceptsllc.com/risk-radar.html. [Accessed: 02-Feb-2016].

[47] Stiki, “RM Studio,” Stiki, 2015. [Online]. Available: http://www.riskmanagementstudio.com/. [Accessed: 02-Feb-2016].

[48] S. de la R. de Sáa, M. Á. Gil, G. González-Rodríguez, M. T. López, and M. A. Lubiano, “Fuzzy rating scale-based questionnaires and their statistical analysis,” IEEE Trans. Fuzzy Syst., vol. 23, no. 1, pp. 1–14, 2015.

[49] S. L. R. Vrhovec, T. Hovelja, D. Vavpotič, and M. Krisper, “Diagnosing organizational risks in software projects: Stakeholder resistance,” Int. J. Proj. Manag., vol. 33, no. 6, pp. 1262–1273, 2015.

[50] A. Rodríguez, F. Ortega, and R. Concepción, “A method for the evaluation of risk in IT projects,” Expert Syst. Appl., vol. 45, pp. 273–285, 2016.

[51] Y. Wang and S. Fu, “A General Cognition to the Multi-characters of Software Risks,” 2011 Int. Conf. Comput. Inf. Sci., pp. 737–739, Oct. 2011.

APPENDIX A: RISK FACTORS

ID	Unsorted Risk Statement	Reference	ID	Unsorted Risk Statement	Reference
1	Large Database Size	[14]	65	Developing Wrong Software Functions	[21], [29]
2	Main Storage Constraint	[14]	66	Developing Wrong User Interface	[21], [29]
3	High Platform Volatility	[14]	67	Gold Plating (changing A Working Software)	[21], [29]
4	Bad Development Schedule	[14]	68	Shortfalls In Outsourced Components	[21], [29]
5	Lack Of Analyst Capability	[14]	69	Shortfalls In Externally Performed Tasks	[21], [29]
6	Lack Of Platform Experience	[14]	70	Real-time Performance Shortfalls	[21], [29]
7	Lack Of Use Of Modern Programming Practices	[14]	71	Bad Traceability	[29]
8	Low Usage Of Software Support Tools	[14]	72	Insufficient Verification And Validation	[29]
9	Lack Of Software Developer Competence	[14]	73	Customer Unsatisfied At Project Delivery	[29]
10	Project NOT Fit To Customer Organization	[5]	74	Risk Reducing Technique Producing New Risk	[29]
11	Lack Of Customer Perception	[5]	75	Catastrophe / Disaster	[29]
12	Project- Resource Conflict	[5]	76	Incorrect Project Size Estimation	[22]
13	Customer Conflict	[5]	77	Project Funding Uncertainty	[22]
14	Lack Of Leadership	[5]	78	Rapid Change Of Job	[22]
15	Definition Of The Program (ambiguity)	[5]	79	Change In Working Circumstances By Management	[22]
16	High Political Influences	[5]	80	Hardware Default Changes	[22]
17	Inconvenient Date	[5]	81	Requirement Postponement	[22]
18	Short Term Solution (lack Of Long Term Solution)	[5]	82	Presence Of High Bugs/errors Count	[22]
19	Lack Of Organization Stability	[5]	83	Technology Change	[22]
20	Lack Of Organization Roles And Responsibilities	[5]	84	Underestimation Of Data Increase Due To Software Success	[22]
21	Lack Of Policies And Standards	[5]	85	Lack Of Design And Development Tool Independence	[22]
22	Lack Of Management Support And Involvement	[5]	86	Risk Of Intruders (hackers, Viruses, Trojan Horse)	[22]
23	Lack Of Project Objectives	[5]	87	Misleading Estimation About Skills Of Workers	[22]
24	Lack Of User Involvement	[5]	88	Lack Of Technical Feedback	[22]
25	Lack Of User Acceptance	[5]	89	Compromise On Profit To Save Name	[22]
26	High User Training Needs	[5]	90	Risk Of Economy Distortion	[22]
27	Large Project Size	[5]	91	Expansion Of Software Requirements	[23]
28	Hardware Constraints	[5]	92	Inaccurate Estimation Of Software Effort	[23]
29	Lack Of Reusable Components	[5]	93	Low Knowledge And Understanding Of Clients Regarding The Requirements	[24]
30	Lack Of Cost Controls	[5]	94	Incorrect Requirements	[24]
31	Lack Of Delivery Commitment	[5]	95	Lack Of Frozen Requirements	[24]
32	Lack Of Requirements Stability	[5]	96	Undefined Project Success Criteria	[24]
33	Requirements NOT Complete And Clear	[5]	97	Conflicting System Requirements	[24]
34	Lack Of Testability	[5]	98	Conflict Between User Departments	[24]
35	Implementation Difficulty	[5]	99	Low Number Of Users In And Outside The Organization	[24]
36	High System Dependencies	[5]	100	Instability Of The Client's Business Environment	[24]
37	Lack Of Response Or Other Performance Factors	[5]	101	Dependency On A Few Key People	[24]
38	High Customer Service Impact	[5]	102	Lack Of Staff Commitment, Low Morale	[24]
39	Data Migration Required	[5]	103	Instability And Lack Of Continuity In Project Staffing	[24]
40	Lack Of Pilot Approach	[5]	104	High Number Of People On Team	-
41	Lack Of Alternatives Analysis	[5]	105	Low Team Diversity	[24]
42	Lack Of Quality Assurance Approach	[5]	106	Lack Of Organizational Maturity	[24]
43	Lack Of Development Documentation	[5]	107	Lack of Project leader's experience	[24]
44	No Use Of Defined Engineering Process	[5]	108	High Extent Of Changes In The Project	[24]
45	Late Identification Of Defects	[5]	109	Excessive Schedule Pressure	[24]
46	Bad Defect Tracking	[5]	110	Inadequate Cost Estimating	[24]
47	Lack Of Or Bad Change Control For Work Products	[5]	111	Poor Project Planning	[24]
48	Problem With Physical Facilities	[5]	112	Ineffective Communication	[24]
49	Problem With Hardware Platform	[5]	113	Improper Definition Of Roles And Responsibilities	[24]
50	Tools Unavailability	[5]	114	Need To Integrate With Other Systems	[24]
51	Bad Project Management Approach / Method	[5]	115	Inadequate Configuration Control	[24]
52	Lack Of Project Management Experience	[5]	116	Low Quality Of Software And Hardware Supplier Support	[24]
53	Bad Project Management Attitude	[6]	117	Excessive Reliance On A Single Development Environment	[24]
54	Lack Of Project Management Authority	[5]	118	High Extent Of Linkage To Other Organizations	-
55	Team Member Unavailability	[5]	119	Resource Insufficiency	[24]
56	Bad Or Low Mix Of Team Skills	[5]	120	Intensity Of Conflicts	[24]
57	Lack Of Experience With Software Engineering Process	[5]	121	Lack Of Control Over Consultants, Vendors ,sub-contractors	[24]
58	Lack Of Training Of Team	[5]	122	Massive User Stress	[22]
59	Lack Of Expertise With Application Area (Domain)	[5]	123	Lack Of Project Delivery Milestones	[22]

60	Development Technology NOT Match To Project	[5]	124	Over-optimistic Technology Perceives	[22]
61	Lack Of Development Technology Experience Of Project Team	[5]	125	Staff Turnover	[22]
62	Immaturity Of Development Technology	[5]	126	Backup Issues	[22]
63	High Design Complexity	[5]	127	Bad Preservation Of Intellectuals	[22]
64	Lack Of Support Personnel	[5]	128	Inability To Secure Confidential Customer Data	-

APPENDIX B: FAILURE MODE QUESTIONS

Questions	**Questions**
How much the users are satisfied with the developed application?	How much the users perceived that the system meets the intended functional requirements?
How much is the overall quality of the developed application?	How much system meets user expectations with respect to ease of use?
How well the system was completed within budget?	How much system meets user expectations with respect to response time?
How good the system was completed within schedule?	How much system meets user expectations with respect to reliability?
How do you rate software defects?	How much the application developed is easy to maintain?