International Journal of Hybrid Information Technology
Vol. 9, No.6 (2016), pp. 11-32
http://dx.doi.org/10.14257/ijhit.2016.9.6.02

Top Fifty Software Risk Factors and the Best Thirty Risk Management Techniques in Software Development Lifecycle for Successful Software Projects

Abdelrafe Elzamly[1*], Burairah Hussin[2] and Norhaziah Md Salleh[3]

[1, 2, 3] Information and Communication Technology, Universiti Teknikal Malaysia Melaka (UTeM)
[1]Department of Computer Science, Al-Aqsa University, Gaza, Palestine
[1]E-mail: Abd_elzamly@yahoo.com{alaqsa.edu.ps}*

Abstract

The concern of this study is to identify software risks and controls in the software development lifecycle. The aim of this study is to rank the software risks factors according to their importance and occurrence frequency based on the data source. The survey questionnaire is used to collect data and method of sample selection referred to as 'snowball' and distribution personal regular sampling was used. The seventy six software project managers have participated in this study who works in the Palestinian software development. Fifty software risk factors in all phases SDLC and thirty risk management techniques were presented to respondents. The results show that all risks in software projects were significant and important in software project manager's perspective. However, the ranking of the importance of the risks is assigned according to it: Analysis, planning, maintenance, design, and implementation. In addition, the top ten software risk factors in software development are selected and used for further analysis such as: Risk13, Risk 14, Risk15, Risk16, Risk11, Risk18, Risk12, Risk50, Risk19, and Risk 9. The concern of this paper the top ten controls are used to model its relationship with the risk, such as: C29, C30, C20, C27, C21, C19, C28, C25, C26, and C23. Software risks can be modelled empirically with risk management control techniques. We recommended applying more studies in software risk management practices with real world companies and building tools to identification and analysis software risks based on quantitative and intelligent techniques.

Keyword: *Software Project, Software Risks, Risk Control Techniques, Software Development Lifecycle (SDLC), software risk management*

1. Introduction

Software development projects still fail to deliver acceptable systems on time and within budget. Due to the involvement of risk management in monitoring the success of a software project, analyzing potential risks, and making decisions about what to do with potential risks, the risk management is considered the planned control of risk. Integrating formal risk management with project management is a new phenomenon in software engineering and product management community. In addition, risk is an uncertainty that can have a negative or positive effect on meeting project objectives. According to Al-Ahmad (2012), there are no studies that identify the risk of incorporating these factors into Software Development Life Cycle (SDLC) [1]. In the process of understanding the factors that contribute to software

* Corresponding Author

project success, risk is becoming increasingly important. This is a result of the size, complexity and strategic importance of many of the information systems currently being developed. In fact, there are many articles were interestingly and describe risk management theoretically, but we need practical models to assess risk and predict risk in software project. Indeed, risk management approach needs more effort from scholars and researchers in quantitative and intelligent risk models [2]. However, the development of software with software risk management methodology is rarely found. Thus, it is important to combine between software life cycle with software risk management such as qualitative, quantitative, and mining techniques to help software manager tracking and mitigate software. The objectives of this study are**:** To identify the software risk factors of software projects and risk control techniques in the Palestinian software development organizations, to rank the software risk factors according to their importance and occurrence frequency based on the data source.

2. Literature Review

Elzamly and Hussin [3] improved quality of software projects of the participating companies while estimating the quality–affecting risks in IT software projects. The results show that there were 40 common risks in software projects of IT companies in Palestine. The amount of technical and non-technical difficulties was very large. In addition [4], we also used new techniques the regression test and effect size test proposed to manage the risks in a software project and reducing risk with software process improvement. Also, they introduced the linear stepwise discriminant analysis model to predict software risks in software analysis development process. These methods were used to measure and predict risks by using control techniques [5]. Additionally, we proposed artifact model of the software risk management for mitigating risks. It has the five levels to mitigate risks through software project [6]. Also, they used the chi-square test to control the risks in a software project [7]. Therefore, the model's accuracy slightly improves in stepwise multiple regression rather than fuzzy multiple regression. However, this methodology based on literature review, the objectives of this paper will achieve followed by survey and discussions with 76 software project managers to estimate the software risk factors and risk management techniques that affect the software project success.

2.1. Software Project

A software project that solution is a functioning software-based information system such as enterprise resource planning system, software package, reports, tools analysis, reengineering software, and website design [8]. Furthermore, increasing demand for new software project is expected to further compound quality risks in software lifecycle [9]. Islam (2009) reported that software project is usually faced with an unexpected problem that is difficult to estimate issues within the software development process. He classified the issues into technical and non-technical during the development of software project [10]. Every software project has challenges which need to be alleviated to make it a successful completion [11]. In addition, the success of software project increasingly important to the survival of business. However, these kinds of software projects are the ones with the highest rate of failure [12]. Risk management projects are increasingly recognized as the practices in the software project organizations for mitigating risks before they occur [13]. Islam (2009) also contributed to a risk management project model to reduce risk at the requirement stage. According to Begum *et al.* [14], a key success for software project factors in software organizations is the software process improvement. Therefore, it is clear that without a good process, a software organization will fail to produce high quality software, mitigating risks and possibly fail to reach its objectives. Such problems in the software process model are missing in the target set for software process and improvement, low involvement of quality control activities, and the absence of standard business expertise practice. Many solutions to enhance software process measurement by tools, techniques, and practices have been suggested [15]. Therefore, it is important to identify the Critical Success Factors (CSFs) that increase the probability of project success. Indubitably, there is a need to focus on software project risk management practice in order to estimate software project risks.

2.2. Software Development Life Cycle (SDLC)

SDLC is a framework that is used to recognize and develop an information system or software project [16]. It is an approach to develop a software project that is characterized by a sequence of steps that progress from the beginning to the end. The SDLC model is one of the oldest systems development model and is still probably the most commonly applied in software development projects [14]. Furthermore, every software project has risks at every stage of the software development lifecycle [17]. A software development life cycle methodology is a structure imposed on the development of a software product [18]. Therefore, there are many methodologies for software development life cycle such as waterfall, V-model, Evolutionary model, spiral development, rapid application development, agile methods, *etc.* as described in Table 2.2. Thus, the agile software development methodology is widely used to collect the values, principles and practices for modelling software in SDLC as well as used to identify and maintain a clear and correct understanding of the software development project being built [19], [20]. Furthermore, it don't contain any risk management techniques because it is believed that short iterative development cycles [21]. The waterfall model is a systematic sequence design process of phases starting with the capture and definition of the requirements, the analysis of these requirements, the formalizing of a system and software design, the implementation of the design, and the testing and maintenance of the software [19]. In particular, the waterfall process model encourages the software development team to specify what the software is supposed to do (gather and define system requirements) before developing the software project [22]. Moreover, the spiral model methodology involves a series of iterations around the requirements capture or specification, implementation, testing, validation, delivery, and operation loop together with periodic reviews of the overall project and the analysis of risks that have been identified during the course of the software project. Rapid applications development and evolutionary delivery are similar sorts of approaches that are built around the idea of building and demonstrating, and in the latter case delivering, parts of the system as the project goes along [19]. The extreme programming (XP) model is used to understand the fundamental values that include its reason for existence and the reason for the successful software project [20]. The V-model is a software development process that can be considered as an extension of the waterfall model. It divides the whole process as verification and validation phases, and each verification phase has a corresponding validation phase. Generally, the component of SDLC consists of planning, analysis, design, implementation, and maintenance. Briefly, the discussion about phases is described below [23]:

- **Planning:** During this phase, the group that is responsible for creating the system must first determine what the system need to do for the organization and evaluation of the existing systems/software.

- **Analysis**: It includes looking at any existing system to see what it does for the organization and how well that system does its job.
- **Design**: It involves the actual creation and design of a system. This involves putting together the different pieces that creates the system.
- **Implementation:** It involves the actual construction and installation of a system.
- **Maintenance:** It includes any future updates or expansion of the system.

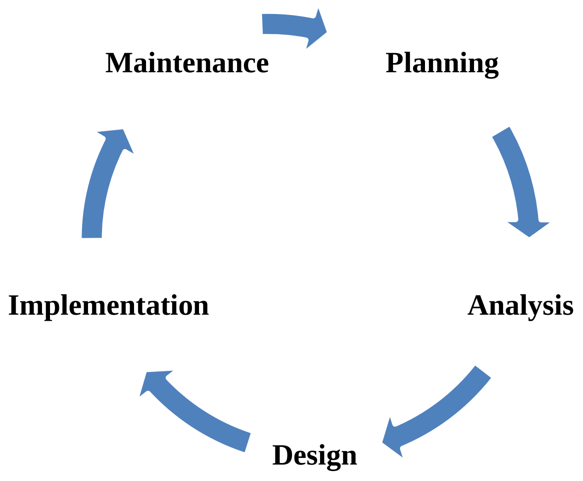


Figure 1. Standard Software Development Life Cycle (SDLC) [23]

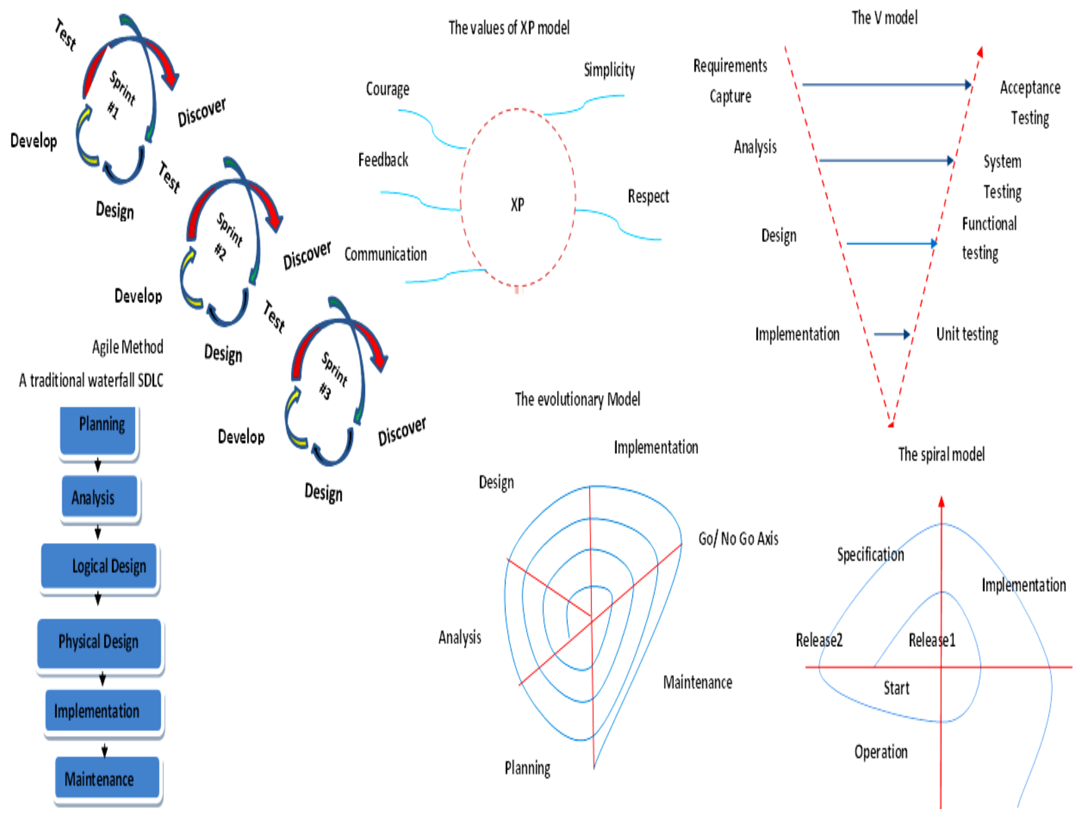


Figure 2. Software Development Life Cycle SDLC methodologies: Waterfall, V- model, Evolutionary model, spiral development, and agile [18–20, 23]

2.3. Software Risk Management

Although there are many methods in software risk management, software development projects have a high rate of risk failure. Thus, if the complexity and the size of the software projects are increased, managing software development risk becomes more difficult [24]. In addition, the optimization method was tested with various software project risk prediction models that have been developed [25]. Following is a discussion on several software risk management approaches, models, and frameworks based on past literature. It is reported that Carr and Tah (2001) have proposed a methodology in software development that covers both process and information system models that are based on the software risk management framework [26]. In terms of economy this methodology provides software managers with a sixth sense that there may be something wrong with the software risk management approach thus enabling them to utilize their knowledge and self-judgment according to their experiences [27]. Fakhar *et al*. (2013) proposed a risk management system based on three risk management steps that include risk identification, risk reduction and risk control [28]. According to Ernawati *et al*. (2012), presented framework for software risk management depends on ISO 31000 and it utilizes a designed architecture that includes the basic components of software risk management like risk identification and risk analysis [29]. Furthermore, Bannerman (2010) postulates that risk management approach practices need to be increased with extra analysis so as to identify, analyze and assess structural risks and to mitigate software risks in software projects [30]. Büyüközkan and Ruan (2010) present incorporated multi-criteria to estimate the methodology for software managers to mitigate software risks. The method relied on a special fuzzy operator, namely a two-additive Choquet integral that enables the modeling of various effects of importance and interactions among software risks [31]. In addition, Oracle Corporation (2010) proposed risk management solutions that enable a standardized approach for identifying, assessing and mitigating risk throughout the software project lifecycle [32]. Dhlamini *et al*. (2009) demonstrated the need for an intelligent risk assessment and management tool for either agile or traditional methods in a software development [33]. Therefore, they proposed a model that could be investigated for use in developing intelligent software risk management tools. Islam (2009) also proposed a Goal-driven Software Development Risk Management Model (GSRM) that supports the identification, assessment, treatment, and documentation of risks in relation to software project-specific goals [10]. Costa *et al*. (2005) proposed a method to measure the possibility for the distribution of harms and earnings that can be incurred by a software development organization according to its software development [34].

Besides, Miler & Górski (2004) proposed a framework modeling the process evolution, which contains techniques to identify process risks and to derive at suggestions for improvement in the software process improvement [35]**.** Padayachee (2002) designed a new software risk management framework by determining the risk performance measure based on a quantitative survey, which was then applied to a risk management strategy [36]. Carr and Tah (2001) posit on a systematic approach to software risk management that involves the identification of risk sources, the quantification of their effects, the development of responses to these risks; and the control of residual risks in the software project estimates. In addition, it was proposed that the principles to manage software project risks by using risk management approach that is proactive, integrated, systematic, and disciplined [37]. Boehm (1991) reiterate that software risk management involves two main phases such as the risk assessment phase that comprise risk identification, risk analysis, and risk prioritization as well as the risk control phase that includes risk planning, risk resolution, and risk monitoring [38]. The approaches and methods reviewed above do not focus on the modelling of software risks based on quantitative and intelligent techniques for predicting the reliability of a software project. Furthermore, there was no integration between the software development life cycle and the real software risk management phases, which were based on techniques to manage software risks. Therefore, it was evident that previous studies for approach in software risk management limited phases and techniques, thus did not create a relation between the software risk factors in software development lifecycle and risk management techniques to mitigate risks. Besides, none of them used the modelling approach to mitigate failure risks in software development. Hence, this study attempts to propose a modelling software risk management for successful software project. On the other hand, the modelling software project for risk management focused on activities that include three factors that are follows as Data source: Questionnaire, historical data, *etc*. Models: Risk stepwise multiple regression modelling, risk fuzzy multiple regression modelling. Methods: Risk identification that rely on risk qualitative models, risk analysis that relies on risk quantitative techniques and risk intelligent techniques, and risk controlling that rely on quantitative and intelligent techniques, *etc*. Unfortunately, quantitative and intelligent techniques are used merely as restrictions in software risk management practice to mitigate risks. However, the software project manager determines the software risk factors and control factors affecting the Software Development Life Cycle phases through the execution of the software projects. Notably, previous studies in software risk management, stress on various phases that must be implemented to mitigate software risks such as risk planning, risk identification, risk prioritization, risk analysis, risk evaluation, risk treatment, risk controlling, and risk communication and documentation [39]. Undeniably, a comfortable model for quantitative risk management approach with software development lifecycle is thus needed. It is applicable to manage risks with stepwise and fuzzy multiple regression analysis techniques. These techniques were used to construct predictive models between risks and controls in the iterative process risk management approach. Furthermore, the display of these phases in Figure 3 is based on the review of literature in above-mentioned section:

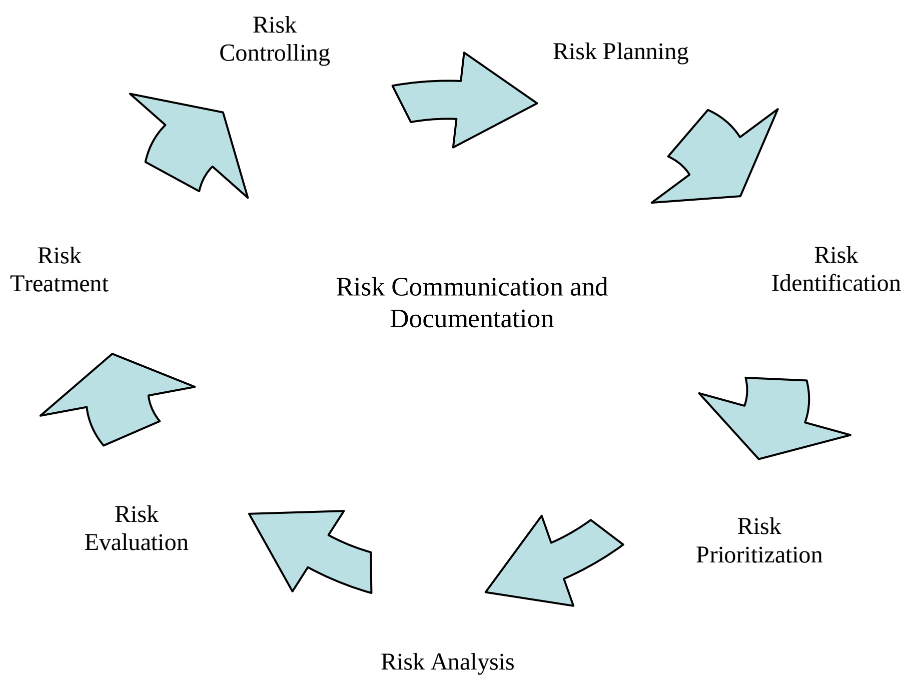


Figure 3. Software Risk Management Phases for Successful Software Project

2.4. Top 50 Software Security Risks in Software Development Lifecycle

This study displays the top 50 software risk factors in software development lifecycle that common in the literature review. The 'Top 10 software risk factors' lists differ to some extent from author to author, but some essential software risk factors that appear almost on any list can be distinguished. These factors need to be addressed and thereafter need to be controlled. Consequently, the list consists of the 10 most serious risks of a software project ranked from one to ten, each risk's status, and the plan for addressing each risk [40]. However, the software risk factors listed in Table 1 below are considered in this study. In addition, these factors are the most common factors used by researchers and experts when studying the software risk factors in software development lifecycle.

Table 1. Illustrate Top Software Risk Factors in Software Project Lifecycle Based on Researchers

Phase	No	Software risk factors	Frequency
Planning[5], [41]–[43]	1	Low key user involvement.	14
	2	Unrealistic schedules and budgets.	14
	3	Unrealistic scope and objectives (goals).	8
	4	Insufficient/inappropriate staffing.	8
	5	Lack of senior management commitment and technical leadership.	8
	6	Poor/inadequate planning.	7
	7	Lack of effective software project management	6

Phase	No	Software risk factors	Frequency
		methodology.	
	8	Change in organizational management during the software project.	5
	9	Ineffective communication software project system.	3
	10	Absence of historical data (templates).	2
	Total frequency		**75**
Analysis[5], [40], [44]	1	Unclear, incorrect, continually and rapid changing software project requirements.	19
	2	Failure to incomplete or missing detailed requirements analysis.	9
	3	Developer software gold-plating.	7
	4	Lack of IT Management.	6
	5	Software project requirements not adequately identified and mismatch.	6
	6	Inadequate knowledge about tools and programming techniques.	5
	7	Lack of traceability, confidentiality, correctness and inspection of the software project planning.	4
	8	Major requirements change after software project plan phase.	3
	9	Changing software project specifications.	2
	10	Inadequate value analysis to measure progress.	2
	Total frequency		**63**
Design[45], [46]	1	Introduction of new technology.	5
	2	Developing the wrong software functions and properties.	5
	3	Developing the wrong user interface.	4
	4	Insufficient procedures to ensure security, integrity and availability of the database.	4
	5	Lack of integrity/consistency.	4
	6	Lack of architecture and quality software project.	3
	7	Absence of quality architectural and design documents.	3
	8	Failure to redesign and design (blueprints) software processes.	2
	9	Lack of effective software project team integration between clients, the supplier team and the supply chain.	1
	10	Misalignment of software project with local practices and processes.	1
	Total frequency		**32**
Implementation[47]–[49]	1	Failure to gain user commitment.	5
	2	Personnel shortfalls.	4
	3	Failure to utilize a phased delivery approach.	2
	4	Too little attention to breaking development and implementation into manageable steps.	2
	5	Inadequate training team members.	1
	6	Inadequacy of source code comments.	1
	7	Inadequate test cases and generate test data.	1
	8	Real-time performance shortfalls.	1
	9	Test case design and Unit-level testing turns out very	1

Phase	No	Software risk factors	Frequency
		difficult.	
	10	Lack of adherence to programming standards.	1
		Total frequency	**19**
Maintenance[50]–[52]	1	Inadequate knowledge/skills.	11
	2	Inadequate change management.	6
	3	Corporate politics with negative effect on software project.	5
	4	Lack of resources and reference facilities.	4
	5	Lack of top management commitment and support and involvement.	4
	6	Shortfalls in externally furnished components, COTS.	3
	7	Legacy software project.	1
	8	Acquisition and contracting process mismatches.	1
	9	User documentation missing or incomplete.	1
	10	Harmful competitive actions.	1
		Total frequency	**37**

2.5. Risk Management Techniques

Through reading the existing literature on software risk management, we listed thirty risk management techniques that are considered important in reducing the software risk factors identified. In the study, we summarize 30 control techniques in mitigating risk as follows[44], [51], [53], [52]: C1: Using of requirements scrubbing, C2: Stabilizing requirements and specifications as early as possible, C3: Assessing cost and scheduling the impact of each change to requirements and specifications, C4: Develop prototyping and have the requirements reviewed by the client, C5: Developing and adhering a software project plan, C6: Implementing and following a communication plan, C7: Developing contingency plans to cope with staffing problems, C8: Assigning responsibilities to team members and rotate jobs, C9: Have team-building sessions, C10: Reviewing and communicating progress to date and setting objectives for the next phase, C11: Dividing the software project into controllable portions, C12: Reusable source code and interface methods, C13: Reusable test plans and test cases, C14: Reusable database and data mining structures, C15: Reusable user documents early, C16: Implementing/Utilizing automated version control tools, C7: Implement/Utilize benchmarking and tools of technical analysis, C18: Creating and analyzing process by simulation and modeling, C19: Provide scenarios methods and using of the reference checking, C20: Involving management during the entire software project lifecycle, C21: Including formal and periodic risk assessment, C22: Utilizing change control board and exercise quality change control practices, C23: Educating users on the impact of changes during the software project, C24: Ensuring that quality-factor deliverables and task analysis, C25: Avoiding having too many new functions on software projects, C26: Incremental development (deferring changes to later increments), C27: Combining internal evaluations by external reviews, C28: Maintain proper documentation of each individual's work, C29: Provide training in the new technology and organize domain knowledge training, C30: Participating users during the entire software project lifecycle.

3. Empirical Strategy (A Case Study)

3.1. Data Collection: Quantitative

Data collection method was achieved using a structured questionnaire for assisting in estimating the quality of software through determining the risks that were common to the majority of software projects in the analyzed software companies. Besides, the method of sample selection referred to as 'snowball' and distribution of personal regular sampling was used. This procedure is appropriate when members of homogeneous groups (such as software project managers, IT managers) are difficult to locate. The seventy six software project managers participated in this case study. The project managements that participated in this survey came from specific, mainly software project managements in software development organizations. Fifty software risk factors and thirty risk management techniques were presented to respondents. The targeted data for this study is undertaken for various software project experts in various software companies in Palestine. There are two data collection process is conducted during the study. The first is a pilot study to validate the instrument to develop during the study, and secondly a mass survey to a target group with the final survey instrument.

3.2. Design of Questionnaire Tools

Respondent was presented with various questions relates to software risks and risk management techniques. The respondents were presented with various questions, which used scales 1-7. For presentation purposes in this paper and for effectiveness, the point scale was the following: For choices, being headed, 'unimportant' equals one and 'extremely important' equals seven. Similarly, seven frequency categories were scaled into 'never' equals one and 'always' equals seven. All questions in software risk factors were measured on a seven–point Likert scale from unimportant to extremely important and software control factors were measured on a seven–point Likert scale from never to always. Therefore, the more extreme categories were combined in a way such that seven- point scales were reduced to five-point scale as follows: A category called 'Somewhat Important' was created, combining the two ratings 'Very Slightly Important' and 'Slightly Important'. Similarly, a category called 'Very Important' combined the two ratings 'very important' and 'Extremely Important'. Similarly, seven frequency categories were rescaled into five subcategories for presentation purposes. 'Rarely' combined the two ratings: 'Rather seldom' and 'Seldom'. 'Never', 'Sometimes' and 'Often' was unchanged, while 'Most of the time', combined the two ratings: 'Usually' and 'Always'. All questions in the software risk factors measure in a seven point Likert scale and risk management techniques also a seven scales, but with different notation that follow in Table 2 below:

Table 2. Measures Scale Software Risks and Controls

Scale	Software risk management	Risk management techniques
1	Unimportant	Never
2	Very Slightly Important	Rather Seldom
3	Slightly Important	Seldom

4	Moderately Important	Sometimes
5	Important	Often
6	Very Important	Usually
7	Extremely Important	Always

The software project managers that participated in this survey are coming from specific mainly software project manager in software development organizations. We identify the software risks that are involved in software projects in Palestinian software development organizations, ranking the risks due to their importance and occurrence frequency, identifying the activities performed by project managers to control the risks that are identified and analyzed. The main survey was sent to the software project manager, IT manager in Palestine organization's individual. Twenty software project managers are working for development software to conduct the pilot survey and 76 to conduct the main survey. The summary of responses for each item from the pilot survey is listed in below. However, the survey questionnaires distributed just the company's top IT manager, software project manager in the software development organizations.

3.3. Pilot Study

A pilot survey questionnaire executed before conducting the main survey questionnaire. The purpose of this pilot survey questionnaire is to examine whether or not the proposed model was well developed to manage software project risks. It is also examined how well the survey is designed for respondents to answer properly. The conceptual managing software project risks and contents of the main survey will be modified depending on the results of the pilot survey. The pilot survey test conducted on software project manager within the population and the feedback received after distributing it to experts in software engineering area, we considered in the pilot survey before sending the main survey and it's available for software project managers, top IT managers more than the experts reviewed it and give us feedback to update an unclear items before sending the main survey to population sample.

3.4. Study Population and Sampling Criteria

The population was all software development organizations in Palestine that have top manager, software project managers. However to describe “Software Development Companies in Palestine” which have in-house software development system and supplier of software for local or international market, we depended on Palestinian Information Technology Association (PITA) Members' web page on PITA's website [http://www.pita.ps/, PITA 2012], Palestinian investment promotion agency [http://www.pipa.gov.ps/, PIPA 2012] to select top IT manager, software project managers in our case study. However, we depend on special criteria to select software companies and participate in our questionnaire by visiting web pages and phone calls before start distributed it.

3.5. Research Instrument Validation and Reliability Pilot Tests

Based upon the pilot study, we believed that the questionnaire is valid and can further use to distribute to the target respondent. For this, 76 software managers for various software companies have participated in the study. The method of sample selection referred to as ‘snowball’ and distribution personal regular sampling was used. This procedure is appropriate when members of homogeneous groups (such as software project managers, IT managers) are difficult to locate. The survey questionnaire provided with covering letter, that explained the aims of our study and the information will secure to encourage higher response. In this section, there are three parts of the survey questionnaire: Information about software project managers; software risk factors; and risk management techniques.

3.6. Construct Validity

To assess the validity of managing software project risks instrument, the correlation was employed and identified five factors in their instrument. Validity tests were performed correlation coefficients between the realize construct were examined. Table 2, 3, 4, 5, 6 and 7 illustrate the correlation between items and total factor planning, analysis, design, implementation, and maintenance. The results reveal that most items are significant at the 0.01 and 0.05 levels 2-tailed except q3, q20, and q36. So the validation of instrument is high, hence the instrument is acceptance except risk3, risk20, and risk36 are no significance, However, we must rewrite these risks to enhance the instrument. Furthermore, it illustrates the correlation among factors and overall risk factors.

Table 3. Correlation between Item and Phases

Phase	Item	Value R	VALUE SIG.
Planning	1	.722	.000**
	2	.697	.001**
	3	.149	.531***
	4	.545	.013*
	5	.846	.000**
	6	.788	.000**
	7	.820	.000**
	8	.520	.019*
	9	.744	.000**
	10	.559	.010*
Analysis	11	.545	.013*
	12	.830	.000**
	13	.579	.007**
	14	.565	.009**
	15	.584	.007**
	16	.609	.004**
	17	.634	.003**
	18	.578	.008**
	19	.753	.000**
	20	.174	.463***
Design	21	.669	.001**
	22	.495	.026*
	23	.865	.000**
	24	.823	.000**
	25	.699	.001**
	26	.601	.005**
	27	.505	.023*
	28	.606	.005**
	29	.559	.010*
	30	.548	.012*
Implementation	31	.709	.000**
	32	.725	.000**

Phase	Item	Value R	VALUE SIG.
	33	.704	.001**
	34	.732	.000**
	35	.732	.000**
	36	.424	.062***
	37	.573	.008**
	38	.749	.000**
	39	.810	.000**
	40	.673	.001**
Maintenance	41	.849	.000**
	42	.558	.011*
	43	.574	.008**
	44	.566	.009**
	45	.716	.000**
	46	.477	.033*
	47	.487	.029*
	48	.470	.037*
	49	.577	.008**
	50	.471	.036*

* Correlation is significant at the 0.05 level (2-tailed).
**Correlation is significant at the 0.01 level (2-tailed).
*** No significance Correlations

Table 4. Correlations among Factors and Overall Risk Factors

FACTOR	Planning	Analysis	Design	Implementation	Maintenance	Total risk factors
Planning	1	.788(**)	.673(**)	.688(**)	.816(**)	.915(**)
Analysis	.788(**)	1	.467(*)	.791(**)	.757(**)	.874(**)
Design	.673(**)	.467(*)	1	.645(**)	.668(**)	.793(**)
Implementation	.688(**)	.791(**)	.645(**)	1	.673(**)	.878(**)
Maintenance	.816(**)	.757(**)	.668(**)	.673(**)	1	.891(**)
Total	.915(**)	.874(**)	.793(**)	.878(**)	.891(**)	1

** Correlation is significant at the 0.01 level (2-tailed). * Correlation is significant at the 0.05 level (2-tailed).

3.7. Instrument Reliability Tests

Reliability can create the stability, consistency of a measuring instrument or tool follow bellow the techniques:

3.7.1. Cronbach's Alpha

In order to assess reliability, the Cronbach's alpha was determined for each factor and total risk factors and risk management techniques. If the Cronbach's alpha is greater than 0.7, the construct is deemed to be reliable.

Table 5. Reliability Tests

Factors		N of items	Cronbach's Alpha
Risk factors	Planning	10	0.836
	Analysis	10	0.789
	Design	10	0.836
	Implementation	10	0.872
	Maintenance	10	0.777
	Total risk factors	50	0.951
Control factors	Planning and requirement techniques	7	0.931
	Communication techniques	5	0.920
	Modeling and tools	18	0.964
	Total Control factors	30	0.973

Table 5 shows that all constructs met the reliability criteria, as the lowest alpha was 0.777. In addition, the reliability coefficient of the scale was established by Cronbach's alpha using the SPSS package; the reliability coefficient resulted by Cronbach's alpha for 20 samples are 0.836, 0.789, 0.836, 0.872, 0.777, 0.951 and 0.973. It is considered to be highly significant at the 0.01 level and this ensures the reliability of the scale.

3.7.2. Two Split Half

Table 6. Spearman-Broun Split Half and Guttman Split Half

Factor		N of items	R	Spearman-Brown
Risk factors	Planning	10	0.733	0.846
	Analysis	10	0.547	0.707
	Design	10	0.471	.640
	Implementation	10	0.589	.741
	Maintenance	10	0.497	0.664
	Total risk factors	50	0.846	0.916
Control factors	Modeling and tools	18	0.936	0.967
	-	-	-	**Guttman Coefficient**
	Planning and Requirement techniques	7		0.883
	Communication techniques	5		0.875`
	Control factors	**30**	**0.814**	**0.898**

Table 6 shown the reliability coefficient resulted by Guttman Split-Half and Spearman-Broun Split Half which they represent highly significant.

3.8. Results and Discussion

3.8.1. The Importance of Risk Factors in Software Development Lifecycle

Table 7 illustrates all respondents indicated that the risk of "Ineffective communication software project system" and "absence of historical data (templates)" were the highest risk factors and important. In fact, the all risk factors in planning phase were important; aggregating the responses resulted in the following ranking of the importance of the listed risks (in order of importance): Risk9, Risk 10, Risk3, Risk1, Risk 6, Risk 8, Risk 7, Risk2, Risk 4, and Risk 5. Furthermore, all respondents indicated that the risk of "developer software gold–plating" was the highest risk factors and very important in analysis phase. In fact, the risk factors for risk number 13, 14, 15, 16, 11, 18, 12 were identified as very important, the risk factors for risk number 19, 17, 20 in descending means were identified as important, aggregating the responses resulted in the following ranking of the importance of the listed risks (in order of importance): Risk 13, Risk 14, Risk15, Risk 16, Risk 11, Risk 18, Risk 12, Risk 19, Risk 17, Risk 20. However, Table 7 also illustrates all respondents indicated that the risk of "introduction of new technology" was the highest risk factors and important in design phase. In fact, all risk factors were important; aggregating the responses resulted in the following ranking of the importance of the listed risks (in order of importance): Risk 21, Risk 22, Risk 24, Risk 25, Risk 27, Risk 28, Risk 23, Risk 26, Risk 30, and Risk 29.

In addition, Table 7 illustrates all respondents indicated that the risk of "Inadequacy of source code comments" was the highest risk factors and importance in implementation phase. In fact, all risk factors important, aggregating the responses resulted in the following ranking of the importance of the listed risks (in order of importance): Risk 36, Risk 33, Risk32, Risk 40, Risk 31, Risk 34, Risk 39, Risk 35, Risk 37, and Risk 38. Also all respondents indicated that the risk of "Harmful competitive actions" was the highest risk factors and important in maintenance phase. In fact, all risk factors were important; aggregating the responses resulted in the following ranking of the importance of the listed risks (in order of importance): Risk 50, Risk 49, Risk 45, Risk 48, Risk 46, Risk 41, Risk 44, Risk 42, Risk 43, and Risk 47. However, the ranking of the importance of phases risks (in order of importance): Analysis, planning, maintenance, design, and implementation.

Table 7. Mean Score for Each Risk Factor in SDLC

Phase	Risk	N	Mean	Std. Deviation	% percent
Planning	R9	76	3.934	0.806	78.684
	R10	76	3.868	0.806	77.368
	R3	76	3.842	0.801	76.842
	R1	76	3.803	0.749	76.053
	R6	76	3.789	0.736	75.789
	R8	76	3.711	0.877	74.211
	R7	76	3.697	0.766	73.947
	R2	76	3.684	0.716	73.684
	R4	76	3.658	0.946	73.158
	R5	76	3.618	0.848	72.368
	Total	**76**	**3.761**	**0.543**	**75.211**
Analysis	R13	76	4.145	.743	82.895
	R14	76	4.092	.819	81.842
	R15	76	4.079	.796	81.579
	R16	76	4.026	.748	80.526
	R11	76	4.026	.588	80.526
	R18	76	4.013	.792	80.263
	R12	76	4	.849	80
	R19	76	3.947	.728	78.947
	R17	76	3.921	.963	78.421
	R20	76	3.895	.793	77.895
	Total	**76**	**4.014**	**0.544**	**80.289**
Desig n	R21	76	3.829	0.737	76.579
	R22	76	3.803	0.633	76.053
	R24	76	3.737	0.772	74.737

Phase	Risk	N	Mean	Std. Deviation	% percent
	R25	76	3.711	0.708	74.211
	R27	76	3.645	0.725	72.895
	R28	76	3.632	0.709	72.632
	R23	76	3.632	0.69	72.632
	R26	76	3.605	0.784	72.105
	R30	76	3.592	0.615	71.842
	R29	76	3.566	0.736	71.316
	Total		**3.675**	**0.451**	**73.5**
Implementation	R36	76	3.671	0.661	73.421
	R33	76	3.658	0.793	73.158
	R32	76	3.632	0.746	72.632
	R40	76	3.553	0.79	71.053
	R31	76	3.553	0.807	71.053
	R34	76	3.513	0.757	70.263
	R39	76	3.5	0.808	70
	R35	76	3.487	0.808	69.737
	R37	76	3.474	0.739	69.474
	R38	76	3.474	0.774	69.474
	Total	**76**	**3.551**	**0.562**	**71.026**
Maintenance	R50	76	3.947	0.781	78.947
	R49	76	3.842	0.731	76.842
	R45	76	3.816	0.761	76.316
	R48	76	3.737	0.822	74.737
	R46	76	3.711	0.78	74.211
	R41	76	3.711	0.708	74.211
	R44	76	3.697	0.8	73.947
	R42	76	3.671	0.839	73.421
	R43	76	3.658	0.758	73.158
	R47	76	3.645	0.778	72.895
	Total	**76**	**3.743**	**0.567**	**74.86**

3.8.2. Ranking of Importance of Risk Factors for Project Managers' Experience

Table 8 shows that most of the risks are very important and important the overall ranking of importance of each risk factor for the three categories of project managers' experience.

Table 8. The Overall Risk Ranking of Each Risk Factor

Phase	Risk	Experience 2-5 years	Experience 6-10 years	Experience >10 years
Planning	R 1	R9	R3	R10
	R2	R1	R10	R1
	R 3	R6	R9	R5
	R 4	R10	R6	R3
	R 5	R3	R1	R9
	R 6	R8	R8	R7
	R 7	R5	R7	R6
	R 8	R2	R2	R8
	R 9	R4	R4	R4
	R 10	R7	R5	R2

Phase	Risk	Experience 2-5 years	Experience 6-10 years	Experience >10 years
Analysis	R 11	R13	R13	R15
	R 12	R12	R14	R14
	R 13	R11	R16	R13
	R 14	R16	R15	R18
	R 15	R18	R17	R17
	R 16	R15	R11	R12
	R 17	R14	R19	R20
	R 18	R19	R18	R19
	R 19	R20	R20	R16
	R 20	R17	R12	R11
Design	R 21	R21	R24	R21
	R 22	R22	R22	R22
	R 23	R30	R21	R25
	R 24	R28	R23	R27
	R 25	R27	R26	R26
	R 26	R24	R25	R28
	R 27	R23	R29	R24
	R 28	R25	R28	R30
	R 29	R29	R27	R29
	R 30	R26	R30	R23
Implementation	R 31	R36	R36	R33
	R 32	R32	R33	R40
	R 33	R33	R31	R39
	R 34	R37	R32	R32
	R 35	R40	R38	R34
	R 36	R35	R40	R31
	R 37	R31	R39	R35
	R 38	R34	R34	R37
	R 39	R38	R37	R36
	R 40	R39	R35	R38
Maintenance	R 41	R50	R50	R50
	R 42	R49	R41	R45
	R 43	R46	R49	R49
	R 44	R48	R45	R46
	R 45	R44	R48	R44
	R46	R42	R47	R48
	R47	R45	R42	R47
	R48	R43	R43	R43
	R49	R41	R44	R42
	R50	R47	R46	R41

3.8.3. Top Ten Software Risk Factors

We selected top ten software risk factors from fifty factors. In fact, all software risk factors in top ten were very important, aggregating the responses resulted in the following ranking of the importance of the listed risks (in order of importance): Risk13, Risk 14, Risk15, Risk16, Risk 11, Risk 18, Risk 12, Risk 50, Risk 19, and Risk 9. Table 9 illustrates the top ten checklists of software risk factors on software projects based on a survey of experienced software project managers.

Table 9. Illustrate the Top Ten Risk Factors

Risk	N	Mean	Std. Deviation	% percent
R13	76	4.145	0.743	82.895
R14	76	4.092	0.819	81.842
R15	76	4.079	0.796	81.579
R16	76	4.026	0.748	80.526
R11	76	4.026	0.588	80.526
R18	76	4.013	0.792	80.263
R12	76	4	0.849	80
R50	76	3.947	0.781	78.947
R19	76	3.947	0.728	78.947
R9	76	3.934	0.806	78.684

3.8.4. Frequency of Occurrence of Controls

Table 10 shows the mean and the standard deviation for each control factors. The results of this study show that most of the controls have used most of the time and often.

Table 10. the Mean Score for Each Control Factor

Control	N	Mean	Std. Deviation	% percent
C29	76	4.408	0.803	88.15789
C30	76	4.368	0.907	87.36842
C20	76	4.184	0.668	83.68421
C27	76	4.171	0.755	83.42105
C21	76	4.171	0.7	83.42105
C19	76	4.158	0.612	83.15789
C28	76	4.158	0.767	83.15789
C25	76	4.132	0.718	82.63158
C26	76	4.118	0.653	82.36842
C23	76	4.105	0.741	82.10526
C22	76	4.092	0.786	81.84211
C18	76	4.079	0.726	81.57895
C10	76	4.079	0.726	81.57895
C17	76	4.066	0.718	81.31579
C24	76	4.066	0.639	81.31579
C8	76	4.066	0.736	81.31579
C5	76	4.053	0.728	81.05263
C11	76	4.039	0.756	80.78947
C15	76	4.039	0.621	80.78947
C9	76	4.039	0.756	80.78947
C14	76	4.013	0.683	80.26316
C7	76	4.013	0.721	80.26316
C16	76	4	0.693	80
C12	76	3.987	0.841	79.73684
C6	76	3.987	0.739	79.73684
C4	76	3.987	0.757	79.73684
C3	76	3.974	0.783	79.47368
C2	76	3.934	0.66	78.68421
C1	76	3.895	0.665	77.89474
C13	76	3.868	0.754	77.36842

4. Conclusions

The results show that all risks in software projects were very important and important in software project manager's perspective, whereas all controls are used most of the time, and often. However, the ranking of the importance of phases risks (in order of importance): Analysis, planning, maintenance, design, and implementation. In particular, top ten software risk factors in software development Lifecycle were very important, aggregating the responses resulted in the following ranking of the importance of the listed risks (in order of importance): Risk13, Risk14, Risk15, Risk16, Risk11, Risk18, Risk12, Risk50, Risk19, and Risk9. In addition, the concern of this study the top ten controls have used most of the time. However, “provide training in the new technology and organize domain knowledge training” is the highest; aggregating the responses resulted in the following ranking of the importance of the listed controls (in order of importance): C29, C30, C20, C27, C21, C19, C28, C25, C26, and C23. To achieve our goals, proposed in this study is identifying risks in software project in software organizations in Palestine. The study population is all software project managers, IT managers in Palestinian software development companies. Software project manager can identify the level of importance and probability of occurrence to mitigate risks through a questionnaire. Meanwhile, the results show that rank of software risk factors and control factors, the importance of the factors. However, we also recommended applying more studies in risk management software practices with real world companies and building tools to identification and analysis risks based on qualitative, quantitative and intelligent techniques. As future work, we will intend to apply these study results on a real-world software project to verify the effectiveness of the new techniques and approach on a software project. Likewise, we can use more techniques useful to manage software project risks such as neural network, genetic algorithm, Bayesian statistics, and so on.

Acknowledgments

The authors would like to thank Faculty of Information and Communication Technology, Universiti Teknikal Malaysia Melaka (UTeM) and Al-Aqsa University, Palestine.

References

[1] W. Al-Ahmad, “Knowledge of IT Project Success and Failure Factors: Towards an Integration into the SDLC”, Int. J. Inf. Technol. Proj. Manag., vol. 3, no. 4, **(2012)**, pp. 56–71.

[2] A. Elzamly and B. Hussin, “Quantitative and Intelligent Risk Models in Risk Management for Constructing Software Development Projects : A Review”, Int. J. Softw. Eng. Its Appl., vol. 10, no. 2, **(2016)**, pp. 9–20.

[3] A. Elzamly and B. Hussin, “Estimating Quality-Affecting Risks in Software Projects”,Int. Manag. Rev. Am. Sch. Press, vol. 7, no. 2, **(2011)**, pp. 66–83.

[4] A. Elzamly and B. Hussin, “Managing Software Project Risks with Proposed Regression Model Techniques and Effect Size Technique”, Int. Rev. Comput. Softw., vol. 6, no. 2, **(2011)**, pp. 250–263.

[5] A. Elzamly, B. Hussin, S. Naser, and M. Doheir, “Predicting Software Analysis Process Risks Using Linear Stepwise Discriminant Analysis : Statistical Methods”, Int. J. Adv. Inf. Sci. Technol., vol. 2015, no. June, **(2015)**, pp. 108–115.

[6] A. Elzamly, B. Hussin, and N. Salleh, “Methodologies and techniques in software risk management approach for mitigating risks: A review”, Asian J. Math. Comput. Res., vol. 2, no. 4, **(2015)**, pp. 184–198.

[7] K. Khanfar, A. Elzamly, W. Al-Ahmad, E. El-Qawasmeh, K. Alsamara, and S. Abuleil, “Managing Software Project Risks with the Chi-Square Technique”, Int. Manag. Rev., vol. 4, no. 2, **(2008)**, pp. 18–29.

[8] J. Miler, “A Method of Software Project Risk Identification and Analysis”, Gdansk University of Technology, **(2005)**.

[9] J. Liu, V. Chen, C. Chan, and T. Lie, “The impact of software process standardization on software flexibility and project management performance: Control theory perspective”, Inf. Softw. Technol., vol. 50, no. 9–10, **(2008)**, pp. 889–896
[10] S. Islam, “Software Development Risk Management Model – A Goal Driven Approach,” in Proceedings of the doctoral symposium for ESEC/FSE on Doctoral symposium, **(2009)**, pp. 5–8.
[11] R. Bhujang and S. V., “Risk Impact Analysis across the Phases of Software Development”, Lect. Notes Softw. Eng., vol. 2, no. 3, **(2014)**, pp. 282–287.
[12] A. Kanane, “Challenges related to the adoption of Scrum”, **(2014)**.
[13] J. Liu, H. Chen, C. Chen, and T. Sheu, “Relationships among interpersonal conflict, requirements uncertainty, and software project performance”, Int. J. Proj. Manag., vol. 29, no. 5, **(2011)**, pp. 547–556
[14] Z. Begum, M. Khan, M. Hafiz, S. Islam, and M. Shoyaib, “Software Development Standard and Software Engineering Practice: A Case Study of Bangladesh”, J. Bangladesh Acad. Sci., vol. 32, no. 2, **(2008)**, pp. 131–139.
[15] T. Dyba and T. Dingsoyr, “Empirical studies of agile software development: A systematic review”, Inf. Softw. Technol., vol. 50, no. 9–10, **(2008)**, pp. 833–859.
[16] R. Dash and R. Dash, “Risk Assessment Techniques for Software Development”, Eur. J. Sci. Res., vol. 42, no. 4, **(2010)**, pp. 629–636.
[17] S. Islam, H. Mouratidis, and E. Weippl, “An empirical study on the implementation and evaluation of a goal-driven software development risk management model”, Inf. Softw. Technol., vol. 56, no. 2, **(2014)**, pp. 117–133.
[18] L. Enfei, “Risk Factors of Software Development Projects in Chinese IT Small and Medium Sized Enterprises”, Kth Royal Institute of Technology ,Stockholm, Sweden, **(2015)**.
[19] M. Holcombe, Running an Agile Software Development Projec*t*. John Wiley & Sons, Inc., **(2008)**.
[20] S. Thomas and U. Hansmann, Agile Software Development: Best Practices for Large Software Development Projects, Springer-Verlag Berlin Heidelberg, **(2010)**.
[21] M. Tomanek and J. Juricek, “Project Risk Management Model Based on Prince2 and Scrum Frameworks”, Int. J. Softw. Eng. Appl., vol. 6, no. 1, **(2015)**, pp. 81–88.
[22] S. Kan, Metrics and Models in Software Quality Engineering, Second. Addison Wesley, **(2002)**.
[23] J. Hoffer, J. George, and J. Valacich, Modern Systems Analysis and Design, 6th ed. Prentice Hall, **(2011)**.
[24] H. Hoodat and H. Rashidi, “Classification and Analysis of Risks in Software Engineering”, Eng. Technol., vol. 56, no. 32, **(2009)**, pp. 446–452.
[25] F. Reyes, N. Cerpa, A. Candia, and M. Bardeen, “The optimization of success probability for software projects using genetic algorithms”, J. Syst. Softw., vol. 84, no. 5, **(2011)**, pp. 775–785.
[26] V. Carr and J. Tah, “A fuzzy approach to construction project risk assessment and analysis: construction project risk management system”, Adv. Eng. Softw., vol. 32, no. 10–11, **(2001)** , pp. 847–857.
[27] C. Pandian, Applied software risk management*:* A guide for software project managers. Auerbach Publications is an imprint of the Taylor & Francis Group, **(2007)**.
[28] M. Fakhar, M. Abbas, and M. Waris, “Risk Management System for ERP Software Project”, in Science and Information Conference 2013, **(2013)**, pp. 223–228.
[29] T. Ernawati, Suhardi, and D. Nugroho, “IT Risk Management Framework Based on ISO 31000:2009”, in International Conference on System Engineering and Technology (ICSET), **(2012)** , pp. 1–8.
[30] P. Bannerman, “Managing Structure-Related Software Project Risk: A New Role for Project Governance”, in 21st Australian Software Engineering Conference, **(2010)**, pp. 129–138.
[31] G. Büyüközkan and D. Ruan, “Choquet integral based aggregation approach to software development risk assessment”, Inf. Sci. (Ny)., vol. 180, no. 3, **(2010)**, pp. 441–451.
[32] Oracle, “A Standardized Approach to Risk Improves Project Outcomes and Profitability”, **(2010)**.
[33] J. Dhlamini, I. Nhamu, and A. Kachepa, “Intelligent Risk Management Tools for Software Development”, in *P*roceeding SACLA “09 Proceedings of the 2009 Annual Conference of the Southern African Computer Lecturers” Association, **(2009)**, pp. 33–40.
[34] H. Costa, M. Barros, and G. Travassos, “A risk based economical approach for evaluating software project portfolios”, ACM SIGSOFT Softw. Eng. Notes, vol. 30, no. 4, **(2005)**, p. 1.
[35] J. Miler and J. Górski, “Identifying Software Project Risks with the Process Model”, in proc. of 17th International Conference “Software & Systems Engineering and their Applications, no. 4, **(2004)**, pp. 1–9.
[36] K. Padayachee, “An Interpretive Study of Software Risk Management Perspectives,” in annual research conference of the South African institute of computer scientists and information technologists on Enablement through technology, **(2002)**, pp. 118 –127.
[37] SQAS, “Software Risk Management A Practical Guide.”, **(2000)**.
[38] B. Boehm, “Software Risk Management: Principles and Practices,” IEEE Softw., vol. 8, no. 1, **(1991)**, pp. 32–40.
[39] A. Elzamly and B. Hussin, “An Enhancement of Framework Software Risk Management Methodology for Successful Software Development”, Journal Theor. Appl. Inf. Technol., vol. 62, no. 2, **(2014)**, pp. 410–423.
[40] A. Elzamly and B. Hussin, “Managing Software Project Risks (Analysis Phase) with Proposed Fuzzy

Regression Analysis Modelling Techniques with Fuzzy Concepts", J. Comput. Inf. Technol., vol. 22, no. 2, **(2014)**, pp. 131–144.

[41] A. Elzamly and B. Hussin, "Managing Software Project Risks (Planning Phase) with Proposed Fuzzy Regression Analysis Techniques with Fuzzy Concepts", Int. J. Inf. Comput. Sci., vol. 3, no. 2, **(2014)**,.pp. 31–40.

[42] A. Elzamly, B. Hussin, S. A. Naser, and M. Doheir, "Classification of Software Risks with Discriminant Analysis Techniques in Software planning Development Process", Int. J. Adv. Sci. Technol., vol. 81, no. 2015, **(2015)**, pp. 35–48.

[43] A. Elzamly and B. Hussin, "Modelling and Evaluating Software Project Risks with Quantitative Analysis Techniques in Planning Software Development", J. Comput. Inf. Technol., vol. 23, no. 2, **(2015)**, pp. 123–139.

[44] A. Elzamly and B. Hussin, "A Comparison of Stepwise And Fuzzy Multiple Regression Analysis Techniques for Managing Software Project Risks : Analysis Phase", J. Comput. Sci., vol. 10, no. 10, **(2014)** , pp. 1725–1742.

[45] A. Elzamly and B. Hussin, "Managing Software Project Risks (Design Phase) with Proposed Fuzzy Regression Analysis Techniques with Fuzzy Concepts", Int. Rev. Comput. Softw., vol. 8, no. 11, **(2013)**, pp. 2601–2613.

[46] A. Elzamly and B. Hussin, "Estimating Stepwise and Fuzzy Regression Analysis for Modelling Software Design Project Risks", Asian J. Math. Comput. Res., vol. 3, no. 4, **(2015)**, pp. 234–241.

[47] A. Elzamly and B. Hussin, "Modelling and mitigating Software Implementation Project Risks with Proposed Mining Technique," Inf. Eng., vol. 3, no. 2014, **(2014)**, pp. 39–48.

[48] A. Elzamly and B. Hussin, "A Comparison of Fuzzy and Stepwise Multiple Regression Analysis Techniques for Managing Software Project Risks : Implementation Phase", Int. Manag. Rev., vol. 10, no. 1, **(2014)**, pp. 43–54.

[49] A. Elzamly and B. Hussin, "Managing Software Project Risks (Implementation Phase) with Proposed Stepwise Regression Analysis Techniques", International Journal on Information Technology (IREIT), vol. 1, no. 5, **(2013)**, pp. 300–312.

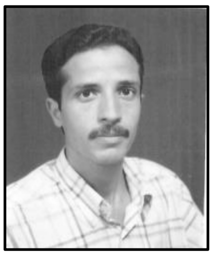
[50] A. Elzamly and B. Hussin, "Evaluation of Quantitative and Mining Techniques for Reducing Software Maintenance Risks", Appl. Math. Sci., vol. 8, no. 111, **(2014)**, pp. 5533–5542.

[51] A. Elzamly and B. Hussin, "Identifying and Managing Software Project Risks with Proposed Fuzzy Regression Analysis Techniques : Maintenance Phase", in 2014 Conference on Management and Engineering (CME2014), no. Cme, **(2014)**, pp. 1868–1881.

[52] A. Elzamly and B. Hussin, "Mitigating Software Maintenance Project Risks with Stepwise Regression Analysis Techniques", J. Mod. Math. Front., vol. 3, no. 2, **(2014)**, pp. 34–44.

[53] A. Elzamly and B. Hussin, "Classification and identification of risk management techniques for mitigating risks with factor analysis technique in software risk management", Rev. Comput. Eng. Res., vol. 2, no. 1, **(2015)**, pp. 22–38.

Authors



Abdelrafe Elzamly, He got a Ph.D. in Information and Communication Technology from the Technical University Malaysia Melaka (UTeM) in 2016. He received his Master degree in Computer Information Systems from the University of Banking and Financial Sciences in 2006. He received his B.Sc. degree in Computer from Al-Aqsa University, Gaza in 1999. He is currently working as Assistant Professor in Al-Aqsa University as a full time. Also, from 1999 to 2007 he worked as a part time lecturer at the Islamic University in Gaza. Between 2010 and 2012 he worked as a Manager in the Mustafa Center for Studies and Scientific Research in Gaza. His research interests are in risk management, quality software, software engineering, cloud computing security, and data mining.

Burairah Hussin, He received his Ph.D. degree in Management Science- Condition Monitoring Modelling, from the University of Salford, UK in 2007. Before that, he received a M.Sc. degree in Numerical Analysis and Programming from the University of Dundee, UK in 1998 and a B.Sc. degree in Computer Science from the University of Technology Malaysia in 1996. He currently works as a Professor at the Technical University Malaysia Melaka (UTeM). He also worked as the Dean at the Faculty of Information and Communication Technology, Technical University of Malaysia Melaka (UTeM). His research interests are in data analysis, data mining, maintenance modelling, artificial intelligence, risk management, numerical analysis, and computer network advising and development.

Norhaziah Md Salleh, She has a M.Sc. in Computing from University of Bradford, United Kingdom in 1993 after getting her B.Sc. in Computer Science from Indiana State University, USA in 1984. She was a systems analyst at Universiti Utara Malaysia from 1986 until 2000. She was a part-time lecturer at Universiti Utara Malaysia for 7 years before joining as a full-time academician in Universiti Pendidikan Sultan Idris (UPSI). She was the Deputy Director of the IT Center at UPSI from 2002 till end of 2003 when she joined Universiti Teknikal Malaysia Melaka as an associate professor. Her research interests include application development, database systems, data quality, systems integration, mobile applications, knowledge management, algorithms, data warehousing and data mining.