Managing Risks in the System Analysis and Requirements Definition Phase

Shihadeh Alqrainy
Albalqa Applied University

Haneen Hijazi
The Hashemite University

ABSTRACT

System analysis and requirements definition is a risky phase. It is susceptible to different types of risk factors from the initial preliminary investigation till the final delivery of the requirements document. Risks reside in this phase are considered the ones with the highest severity among other phases. Being the first phase in the development process, the occurrence of risks in this phase negatively influences subsequent phases, affects project progress, and has a negative impact on the project outcomes. Thus, managing probable risks in this phase deadly helps project managers control the majority of risks that might arise later in the subsequent phase. In order to manage risks properly, probable risks need to be identified early, then, risk management strategies have to be proposed and followed in order to avoid and mitigate their occurrence. In this paper, a total number of 28 risk factors have been introduced. For each risk factor, a set of management strategies is proposed. The identified factors and strategies were the harvest of brainstorming sessions with senior software practitioners, comprehensive literature survey, plus ready-made checklist and taxonomies. In order to validate our results, correlation analysis had been conducted through a web-based survey. The results confirmed our assumptions in that all of the identified risk factors have positive correlation with project failure.

Keywords

Requirements, Risk Factor, Risk Management, Software Development Lifecycle (SDLC)

1. INTRODUCTION

Each software system passes by a sequence of phases before it could be used by end-users. Each phase uses the output of the preceding one. Together, they form what is known the Software Development Life Cycle (SDLC). Respectively, these phases are Systems analysis and requirements definition, design, development, integration and testing, and the deployment and maintenance phase.

System analysis and requirements definition is the first phase which represents the "what" phase and definitely the most important one. Herein, system goals and users needs from the system are clarified.

Software System analysis is the process of studying a software project in order to identify its goals and purposes and create systems and procedures that will achieve them in an efficient way [1]. By analyzing the system, analysts should decide whether the system underdevelopment is feasible to implement especially from economical point of view. System analysis is much related to what is called requirements definition. Requirements definition examines what the stakeholders need without any clarification of how these requirements could be developed [2]. Software requirements can be functional and non-functional. Functional requirements consider the behavioural aspects of the system, while the non-functional concern with the operational aspects. Defining requirements involves elicitation, analyzing, and documenting them [3].

The final output of this phase is the requirements document (RD). Requirements document writes down software requirements specification (SRS) either in natural language or in formal language. In this document, requirements document is specified at higher level; architectural details are left to the subsequent design phase [4].

Starting from the requirements phase and ending with the final acceptance of the system, each phase of the SDLC is vulnerable to several threats [5]. As any other phase in the SDLC, the system analysis and requirements definition phase suffers from a set of probable threats that hinder the successful completion of the project. Indeed, vulnerabilities in this phase are very severe; any fault in describing the requirements may cause other threats appear later in subsequent phases. Worse, it may cause user unsatisfaction, resources underestimation, and consequently entire project failure. These threats are called **risk factors**.

Hence, risks in this phase need to be managed carefully in order to mitigate its negatives effects in the future. This management requires these risks to be identified, and then risk management strategies to be proposed and applied to avoid and mitigate from these risks.

Risk Identification is the step that aims at discovering, as many as possible, potential threats and uncertainties that could compromise project success. The result of the risk identification step is a list of risk factors (i.e. technical, environmental, managerial, and organizational) that could lead to project failure [6]. Many risk identification techniques exist; the most currently practiced are checklist and brainstorming [7]. Risk factors are the uncertain conditions and influences that affect the cost, duration, and quality of the project negatively. The purpose behind the identification process is not the recognition of these risk factors solely. Rather, to help project managers and developers manage these risks.

A risk management strategy is a control activity that aims at dealing with a specific risk factor(s). Not all risk factors are controllable [8]; some factors might be out of project manager’s control. Any software risk factor can be either avoidable or non-avoidable. For the avoidable risk factors, mitigation strategies are devised and proposed to deal with risks before they mature into real problems. Else, if the risks are non-avoidable, or if the risks have matured into real problems, then contingency plans have to take place in order to repair from the occurrence of these risks. A mitigation strategy aims at either avoiding the occurrence of a risk, or reducing its effects in case of occurrence. This reduction can be achieved by reducing either the severity of the risk or its likelihood.

Either the mitigation strategies or the contingency plans must be planned in advance [9]. In other words, avoided the occurrence of the risks has a priority rather than start to think of and design strategies. Clearly, applying a mitigation strategy is better than conducting a contingency plan, since it is cheaper, and easier than repairing from risk. A risk management strategy can control more than one risk factor. As mentioned in the previous sections, the occurrence of a risk might be as a consequence of another risk, thus, mitigating the cause may also mitigate the consequence.

In this paper, we investigate the system analysis and requirements definition phase and identify the major risk factors threaten this phase and its activities. For each factor, a set risk management strategies is proposed to manage that risk. Most of these strategies are avoiding strategies, the other are mitigating strategies. The rest of this paper is structured as follows: section 2 summarizes the works related to this research. In section 3, the identified risk factors and the proposed risk management strategies are introduced. Section 4 validates our assumptions and analyzes the results of the questionnaire. Finally, the conclusion and the future work is shown in section 5.

2. RELATED WORK

Several research works have been conducted around risk management. These attempts are summarized in this section as follows:

Keil et al. in [10], identified and prioritized eleven risk factors. The authors found that there is a relation between the importance of risks and their perceived level of control. Building upon this, they introduced a framework for classifying risks. Four categories were identified; customer mandate (high importance, low control), scope and requirements (high importance, high control), execution (lower importance, high control), and environment (lower importance, lower control). Instead of mitigating individual risk factors, mitigation strategies were devised to handle each type of risks.

Vallabh and Addison [11] reported on risk factors and controls from the literature. This list of factors and controls were presented to different project managers from the IT and finance industry with different experiences, then they were asked to identify the importance of each risk factor, the frequency of occurrence for each risk factor and control, and the effectiveness of each control against each factor. They found that among the identified ten factors, seven were reduced using the using the identified controls.

Shahzad and Safvi [9] presented a list of risk factors and another one of mitigation strategies to a representative set of students, academics, and professionals in order to assign the correct mitigation strategy to each specific factor. They used the list of eighteen risk factors previously identified by Shahzad and Iqbal in 2007 [12]. For each risk factor a set of risk management strategies was proposed. Some of these strategies are mitigation strategies aimed at avoiding, or at least reducing, the related factor. Others are contingency plans take place when the risk is matured into a problem.

Shahzad et al. in 2010 [13] prioritized the risk factors identified earlier by Shahzad and Safvi in 2008 [14] according to the overall impact for each factor. Then the authors suggested that risk factors with highest values have to be addressed first using the defined sets of mitigation strategies. These factors were classified into avoidable and non-avoidable factors based on their priorities. Then for the avoidable risks, a set of mitigation strategies were put in order to avoid them. For the non-avoidable, a set of contingency plans are set for each risk factor in case if it matured into a problem.

Hijazi et al. in 2014 [5], identified a comprehensive list of risk factors that threaten the software development process.

3. THE PROPOSED RISK MANAGEMENT STRATEGIES

In this section the risk factors that threaten the requirements analysis and definition phases are explored. For each factor, a set of mitigation strategies is proposed to help mitigate that factor.

3.1 Factor 1: Inadequate estimation of project time, cost, scope and other resources

Idealistic view of team capabilities, hidden unknown variables, and the human desire to good news leads to unrealistic project schedule, budget, unclear scope, and insufficient resources, which are considered the major project failure causes. Moreover, project managers may find it difficult to estimate the required time, cost, scope and other resources needed to complete the project.

3.1.1 The proposed strategies:

- Consult Information Sources (i.e. different stakeholders).
- Use experience results from previous projects.
- Use different estimates from different sources [13].
- Estimate the amount of the available reusable code adequately.
- proactive estimating techniques

3.2 Factor 2: Unrealistic Schedule

Unrealistic Schedule is a major cause of project failure. The schedule is considered unrealistic if the estimated time for the project as a whole exceeds the delivery date agreed upon previously. This might be due to inexperienced project managers, continually changing requirements, and improper management. Unrealistically, project managers add constraints on time, and overload the developers to deliver on time to mitigate this issue.

3.2.1 The proposed strategies:

- Add flexibility to the schedule, by adding extra contingency factor to the estimated time [15].
- Setup clear milestones.
- Use different time estimation models (i.e. COCOMO, Putam's).
- Estimate the amount of the available reusable code adequately [15].
- Propose changes to the schedule.

3.3 Factor 3: Unrealistic Budget

Unrealistic estimation of cost causes budget runs out early in the SDLC. To estimate the budget project manager should take the required time, effort, and resources into consideration [9, 11]. Any improper estimation in this regard may cause project failure.

3.3.1 The proposed strategies

- Engage project stakeholders in interactive discussion, case studies and practical exercises to help them accurately judge the impact of expenses and make forecasts on project performance.
- Propose changes to the budget to accommodate with emergent changes.

3.4 Factor 4: Unclear project Scope

Project managers should clearly define what the project is supposed to do. This assures that core functionalities are neither missed nor extra ones be taken into consideration.

3.4.1 The proposed strategies:

- Use estimation tools/matrices to determine the exact scope.
- Engage developers, analysts and other team members in the determining project scope [15]
- Estimate the amount of the available reusable code adequately [15].

3.5 Factor 5: Insufficient resources

An accurate estimation of the required resources and the available ones should be made early to make sure that system could be implemented using the available and to avoid technology change.

3.5.1 The proposed strategies:

- Do not impose a project of new technology and limited resources [15].
- Experienced developers should advice the customer with the most suitable tools and technologies [13].

3.6 Factor 6: Unclear Requirements

If the requirements are unclear, this could lead to misunderstandings between the different system's stakeholders. This usually causes a dead lock where the system needs to rebuild again.

3.6.1 The proposed strategies:

- Structure user requirements.
- Held SCRUM meetings

3.7 Factor 7: Incomplete Requirements

Practically, users can describe at most 60% of the requirements at the beginning of the project. Thus, new requirements could arise in later phases. Anyway, any miss in users needs results in incomplete requirements.

3.7.1 The proposed strategies

- Hold structural interviews and Joint Application Development (JAD) workshops [9].
- Use Facilitated Application Specification Technique (FAST) that helps in collecting understanding requirements in an informal way.
- Use hands-on experience.
- Examine existing software.

3.8 Factor 8: Inaccurate Requirements

Accurate requirements should express real user needs exactly and accurately.

3.8.1 The proposed strategies:

- Hold structural interviews and Joint Application Development (JAD) workshops [9].
- Use Facilitated Application Specification Technique (FAST) that helps in collecting and understanding requirements in an informal way.
- Use hands-on experience.
- Examine existing software.

3.9 Factor 9: Ignoring the Non-functional requirements

Non-functional requirements such as system usability, maintainability, scalability, testability are given little attention comparing with the functional requirements which concentrate on what the system should do rather than how the system behaves.

3.9.1 The proposed strategies:

- Hold structural interviews and Joint Application Development (JAD) workshops [9].
- Use Facilitated Application Specification Technique (FAST) that helps in collecting understanding requirements in an informal way.
- Use hands-on experience.
- Examine existing software.

3.10 Factor 10: Conflicting User Req

Conflicting requirements are inevitable in systems with several different users and different needs. Each user wants to see all requirements implemented successfully regardless of others need.

3.10.1 The proposed strategies:

- Hold structural interviews and Joint Application Development (JAD) workshops [9].

3.11 Factor 11: Unclear Description of the real environment

Analysts should have a clear knowledge about the real environment wherein the system will operate. Indeed, it's not an easy task.

3.11.1 The proposed strategies:

- Hold structural interviews and Joint Application Development (JAD) workshops [9].
- Apply Ethnography Techniques; which is an observational technique that can be used in the organizational environment [2].

3.12 Factor 12: Gold Plating

Adding extra functionality to the system that is not considered in the original scope in order to make the system better may cause in most cases a threat to the project as much as, if it was not more than, omitting the in-scope functionalities [4, 11].

3.12.1 The proposed strategies:

- Remove any extra functionality that will not be used.

3.13 Factor 13: Non-verifiable Req

The requirement is non-verifiable if there is not a finite cost effective process (i.e. testing, inspection, demonstration, or analysis) assuring that the software meets the requirements [2].

3.13.1 The proposed strategies:

- Re-write requirements in a verifiable (i.e. quantifiable and measurable) structure.

3.14 Factor 14: Infeasible Requirements

Infeasible requirements are the requirements that have insufficient resources for its implementation within the project's constraints.

3.14.1 The proposed strategies:

- Do not impose a project of new technology and limited resources [15].

3.15 Factor 15: Inconsistent Requirements

Inconsistent requirements are the requirements with any contradictions.

3.15.1 The proposed strategies:

- Hold structural interviews and Joint Application Development (JAD) workshops [9].
- Prioritize requirements.

3.16 Factor 16: Non-traceable Req

Non-traceable requirements cannot be referenced later on easily. This contradicts with documentary and referencing purposes.

3.16.1 The proposed strategies:

- State the source of each requirement.
- Build the traceability matrix.

3.17 Factor 17: Unrealistic Requirements

Realistic requirements are clear, verifiable, accurate, consistent, complete, and feasible to be implemented requirements.

3.17.1 The proposed strategies:

- Since the unrealistic requirements are those requirements that are clear, verifiable, accurate, consistent, complete, or feasible, then all the strategies that are devised to mitigate from these risks are required to mitigate from the unrealistic requirements risks.

3.18 Factor 18: Misunderstood domain-specific terminology

Application specialists and developers use domain-specific terminologies that are different and not understandable by most end-users. This might lead to misunderstanding between both parties [4].

3.18.1 The proposed strategies:

- Formal requirements review [4].
- Develop exploratory prototype.

3.19 Factor 19: Mis-expressing user requirements in natural language

Different users may use different natural languages and different conventions. Moreover, many expressions and terms need more formal language to be expressed.

3.19.1 The proposed strategies:

- Express requirements using structured English.

3.20 Factor 20: Inconsistent requirements data and RD

Any fault in the gathering and documenting technique may cause inconsistency between the actual requirements and the corresponding documented ones.

3.20.1 The proposed strategies:

- Automated-processing of requirements using automated tools to generate the requirement document directly from requirements.

3.21 Factor 21: Non-modifiable RD

Ignoring maintainability while documenting requirements makes it difficult to modify data system usability, maintainability, scalability, and testability.

3.21.1 The proposed strategies:

- Enhance changeability in the RD by organizing the document in an easy to use way.
- Avoid redundancy in the requirements document.

3.22 Factor 22: Continually changing req

In practice, it is difficult or even impossible to describe all requirements at the initial stages of the project, thus, definitely requirements change over the SDLC. To accommodate with this continuous changes, additional work have to be done which might consume extra time and cost. Beside, test plans are designed early according to the initial requirements; hence, testing could not cope with this continuous change.

3.22.1 The proposed strategies:

- Add flexibility to the schedule, by adding extra contingency factor to the estimated time [2].
- Clarifying that large change in requirements be done on additional payments [15].

3.23 Factor 23: Project Funding Loss

Lack of commitment from the funding agencies is another major factor. If the funding was interrupted at any phase, the project could not be completed.

3.23.1 The proposed strategies:

- Get an adequate commitment from the funding agencies in the beginning of the project [15].
- Maintain a friendly relationship with funding agencies [15].

3.24 Factor 24: Team Turnover

Unstable employment and better job vacancies threatens most organizations. Project team members' resignation suddenly negatively affects the development process.

3.24.1 The proposed strategies:

- Give High salaries.
- Give additional payments and rewards for extra, well-done work.
- Hold Training sessions and courses on the latest technologies.
- Establish good relationships with the employees, and between the employees themselves.

3.25 Factor 25: Data Loss

Natural disasters, viruses and intruders, developers run away with codes are all reasons of data loss.

3.25.1 The proposed strategies:

- Data backup must be taken regularly and at different sites.
- Use anti-viruses and firewalls systems.
- Hire highly trusted team members.

3.26 Factor 26: Time contention

Indeed, time contention is not a serious threat in this phase. It usually appears in the implementation phase. Nevertheless, the causes behind this contention reside herein.

3.26.1 The proposed strategies:

- Reuse components.

3.27 Factor 27: Miscommunication

Miscommunication between different project stakeholders may lead to different troubles. For instance, customers may under or overestimate their expectations and developers may not understand the user actual needs.

3.27.1 The proposed strategies:

- Adapt effective communication between stakeholders along the testing process

3.28 Factor 28: Budget Contention

As is the case in the time contention factor, budget contention appears in the later phases of the development process when almost the entire allocated budget for the project is spent. Nevertheless, requirements phase may suffer from this risk factor when unrealistic estimation for budget is made from the beginning and wherein funding agencies suffer uncertainty.

3.28.1 The proposed strategies:

- Allocate a realistic budget for the project from the beginning plus extra amount.
- Contact with stable funding agencies.

4. CORRELATIONS ANALYSIS

In order to validate our assumptions about the just identified risk factors, a correlation analysis has been conducted. Correlation analysis is a statistical measurement of the relationship between two variables. Correlation between two variables does not necessary implies causality. The correlation can be positive, negative, or no correlation. A positive correlation implies that the values of both variables increase or decrease together (goes in the same direction). Negative correlation Indicates that as the value of one variable increases, the other decreases (and vice versa) (goes in opposite directions). No correlation means that there is no relationship between the two variables [16].

Different types of correlation analysis research methods exist; naturalistic observations, survey method, and the archival research. Survey method is the most common method since it is the fastest, cheapest, and the most flexible one. In this method, a random sample of participants completes a survey, test, or questionnaire that relates to the variables of interest [16].

In this context, we used survey method correlation analysis in order to examine the relationship between each of the identified risk factors and project failure. To achieve this, a web-based questionnaire was circulated among different software practitioners using convenience sampling method. The sample was chosen randomly to ensure the generalization of the survey result.

In the questionnaire, all of the 28 identified risk factors were presented to a large sample of people via email and other social networks, and they were asked to describe the relationship between each risk factor and the project failure (i.e. perfect positive, positive, no effect, negative, perfect negative).

A total number of 60 responds were returned. After analyzing the responses, data from respondents were summarized in table 1. In this table, for each of the identified risk factor, the percentage of respondents regarding each correlation analysis value is stated.

Clearly, all the respondents exhibited similar tendencies in which all the identified risk factors have non-negative relationships with project failure. They justified their responses in that all of them will cause loss in time, budget, or quality of the system and thus lead to project failure. These results largely strengthen our assumptions; even that the sampling method followed does not allow us to generalize our results.

Table 1: Analyzing Questionnaire Data – Correlation Analysis between each risk factor and project failure
P-pos = perfect positive, **p-neg** = perfect negative

Risk factor	P-pos	Pos	No eff	Neg	p-neg
Inadequate estimation of project time, cost, scope and other resources	90%	10%	0%	0%	0%
Unrealistic Schedule	55%	45%	0%	0%	0%
Unrealistic Budget	80%	20%	0%	0%	0%
Unclear project Scope	65%	35%	0%	0%	0%
Insufficient resources	65%	35%	0%	0%	0%
Unclear Requirements	70%	30%	0%	0%	0%
Incomplete Requirements	55%	45%	0%	0%	0%
Inaccurate Requirements	50%	50%	0%	0%	0%
Ignoring the Non-functional	30%	65%	0%	0%	0%

requirements					
Conflicting User Requirements	50%	50%	0%	0%	0%
Unclear Description of the real environment	65%	30%	5%	0%	0%
Gold Plating	50%	40%	10 %	0%	0%
Non-verifiable Requirements	35%	60%	5%	0%	0%
Infeasible Requirements	65%	35%	0%	0%	0%
Inconsistent Requirements	65%	30%	5%	0%	0%
Non-traceable Requirements	55%	30%	0%	0%	0%
Unrealistic Requirements	45%	50%	5%	0%	0%
Misunderstood domain-specific terminology	55%	45%	0%	0%	0%
Mis-expressing user requirements in natural language	65%	35%	0%	0%	0%
Inconsistent requirements data and RD	65%	35%	0%	0%	0%
Continually changing requirements	70%	30%	0%	0%	0%
Project Funding Loss	80%	20%	0%	0%	0%
Team Turnover	75%	25%	0%	0%	0%
Data Loss	65%	35%	0%	0%	0%
Time contention	65%	30%	5%	0%	0%
Miscommunication	55%	40%	5%	0%	0%
Budget Contention	80%	20%	0%	0%	0%

5. COMPARITIVE ANALYSIS

Shareef Islam [17] proposed Software Development Risk Management Model- a goal-driven approach, the author's proposed common project riskiness factors as shown in figure 1.

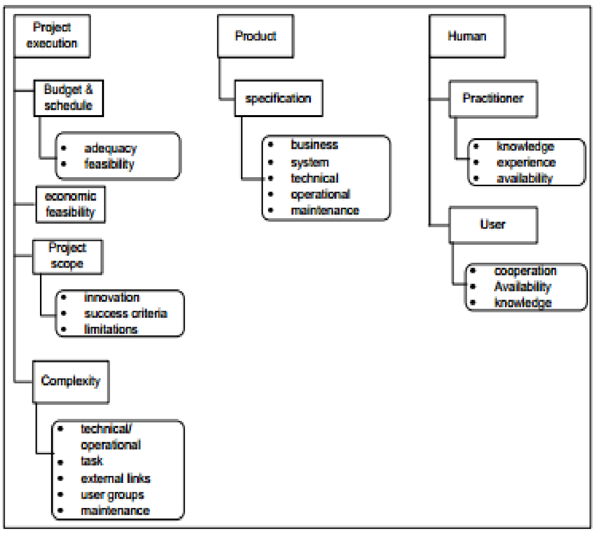

Fig 1: Common project riskiness factors

List of risk event and associated factors that Shareef Islam [17] proposed shown in Table 2.

Table 2: Islam's List of risk event and associated factors

Component	Risk event	Risk factors
Project execution	Budget overruns	Unrealistic cost estimation, hidden factors, schedule overruns, operational and maintenance difficulties, highly complex project, unclear scope, high risky project.
	Schedule overruns	Inaccurate schedule estimation, unclear milestones, project complexity, erroneous requirements , unclear system vision, inadequate activity, incompetence practitioner, user lack of cooperation, numerous change, unclear scope, rework of deliverables.
	Project complexity	High level technical complexity, complex tasks and dependencies with among system components, new and unknown technology, high level innovation, immature technology, long duration, unrealistic expectations.
Process	Inadequate development activity	Inadequate tasks and methods, complex process, unclear roles and responsibilities within the process, untrained practitioner, not followed or partially followed in the project, inconsistency among the artifacts.
Product	Erroneous requirements	Unclear goals, requirement faults i.e., ambiguous, incorrect, unstable, incomplete, immeasurable, over specified, not prioritized and traceable requirements, documentation error, lack of knowledge, inadequate time and budget for RE, passive user involvement, practitioners lack of knowledge, unclear scope.
	Poor quality	Unclear scope and quality goals, erroneous requirements, lately considering quality issues, missing user expectations, lack of analysis of project domain specific quality properties.
	Operational dilemma	Inadequate user and technical manual, unsatisfactory training, lack of training budget, lack of user motivation, incomplete or incorrect data conversion, not harmonized transition plan, incomplete operation specification.
Human	Poor team performance	Frequent conflicts, negative team attitude, lack of motivation, unbalanced team, lack of coordination, unclear roles & responsibilities, incompetence staff
	Passive user involvement	Wrong user representative, inadequate domain knowledge, lack of motivation for the new system, lack of IT competence, poor feedback.
	Incompetence staff	Lack of domain knowledge, unskilled/ untrained staff, inexperience project manager, lack of management support, lack of motivation and productivity.
Environment	Unstable organization	Unstructured organization, management lack of support, inefficient decision making capability, not willing to provide additional budget, political bias, inadequate policies and process.

Determining the project risk factors is not an easy task due to the nature of the project. Shareef Islam [17] developed a model and determined the project risks based on the project benefits at the end. As Shareef Islam [17] claimed that every project has a common goal to not lose any money. Therefore, the author decided how risky the project is in terms of cost, schedule and other related factors as well as shown in table 2. While the proposed model in this paper described a comprehensive risk factors affect the failure of the project during the analysis and requirements phase only as described in table 1. However, a comprehensive study about the project risk factors covered all the project phases presented in [5].

5. CONCLUSION AND FUTURE WORK

In this paper a detailed list of risk factors that threaten the system analysis and requirements definition phase have introduced. In addition, a set of mitigation strategies was defined for each risk factor to help developers and projects managers deal with these risks. In order to validate our assumptions, a correlation analysis has been performed to verify the relationship between the identified risk factors and project failure.

In the future, similar studies could be conducted on other SDLC phases. Moreover, the proposed sets of factors and mitigation strategies may constitute the main building block to develop preventive risk management models, and to integrate risk management into the Software development process.

6. REFERENCES

[1] Harry J. *Systems Analysis and Design.* 10th. Boston : Rosenblatt, (2014).

[2] I, Sommerville. *Software Engineering.* 9th. USA : Addison Wesley, (2011).

[3] K. E. Wiegers. *Software Requirements.* 2nd. Washington : Microsoft Press, (2003).

[4] Board for Software Standardisation and Control. *Guide to the user requirements definition.* ESA, 1995.

[5] H. Hijazi, S. Alqrainy, H. Muaidi, and T. Khdour. *Risk Factors in Software Development Phases.* European Scientific Journal, Vol. 10, No. 3, pp. 213-232, (2014).

[6] J. Dhlamini, *I. Nhamu, and A.* Kachepa, *Intelligent Risk Management Tools for Software Development.* Proceedings of the 2009 Annual Conference of the Southern African Computer. pp. 33-40, (2009) June 29-July 1. Eastern Cape, South Africa.

[7] J. Miler and J. Górski. *Risk-driven Software Process Improvement - A Case Study*. Proceedings of the 11th European Software Process Improvement Conference. (2004) November 10-12. Trondheim, Norway.

[8] S. Zardari . *Software Risk Management.* Proceedings of the 3rd International Conference on Information Management and Engineering. pp. 375-379. (2009). Kuala Lumpur, Malaysia.

[9] B. Shahzad, and S. A. Safvi. *Risk Mitigation and Managemen Scheme based on Risk Priority*. Global Journal of Computer Science and Technology, Vol. 10, No. 4, pp. 108-113, (2010).

[10] M. Keil, P. E. Clue, K. Lyytinen, *and R. S.* Schmidt. *A framework for Identifying Software Project Risks.* Communications of the ACM, Vol. 41, No. 11, pp. 76-83, (1998).

[11] T. Addison and S. Vallabh. *Controlling Software Project Risks: An Empirical Study of Methods used by Experienced Project Managers.* Proceedings of SAICSIT. pp. 128-140. (2002) September 16-18. South Africa

[12] B. Shahzad and J. Iqbal. *Software Risk Management Prioritization of Frequently Occurring Risk in Software Development Phases using Relative Impact Risk Model.* Proceedings 2nd International Conference on Information and Communication Technology. pp. 110-115. (2007) December 16-17. IBA, Karchi.

[13] B. Shahzad, A. S. Al-Mudimigh. Ullah. *Risk Identification and Preemptive Scheduling in Software Development Life Cycle.* Global Journal of Computer Science and Technology, Vol. 10, No. 2, pp. 55-63, (2010).

[14] B. Shahzad and S. A. Safvi. *Effective Risk Mitigation: A User Prospective.* International Journal of Mathematics and Computers In Simulation, Vol. 2, No. 1, pp. 70-80, (2008).

[15] B. Shahzad, I. Ullah, and N. Khan. *Software Risk Identification and Mitigation in Incremental Model.* Proceedings of the International Conference on Information and Multimedia Technology. pp. 366-370. (2009) December 16-18. Jeju Island, Korea.

[16] BIBLIOGRAPHY \l 1033 Cherry, K. (n.d.). *Correlational Studies.* Retrieved May 25, 2014, http://psychology.about.com/od/researchmethods/a/correlational.htm

[17] Islam, Sh. *Software Development Risk Management Model – agoal – driven approach.* Phd thesis. Technische Universit Munchen : Germany, (2011).