Karpagam JCS Vol. 2 Issue 6 September. - October. 2008

Managing Software Risk A Lifecycle Approach

R. A. Khan[1], K. Mustafa[2]

ABSTRACT

In a technology driven area of study, software project risk management has become relevant over time and indicates the importance and complexity of the issue they are addressing. The major objectives of software risk management includes identifying, addressing, and eliminating potential elements of risk before they become either threats to successful software operation or major sources of software rework. In order to develop risk free software, it appears to be worthwhile to incorporate risk management within the development process as part of project management. A process of risk management consisting of six phases including identification, analysis, prioritization, planning, resolving and monitoring has been introduced in this paper. In order to prevent project runaways, meet deadlines, stay within the project's budget, and simultaneously maintain the product's high quality standards, it is essential to timely identify, analyze and mitigate risks associated with each and every phase of software development life cycle. For that, a framework has been proposed to integrate risk within the development life cycle.

1. INTRODUCTION

No doubt, for the successful completion and maintenance of software projects, there is a need for effective software project managers. The long proclaimed ineffectiveness of software development projects to maintain their schedule, cost, and quality, continues to plague most development projects [1, 2, 3]. It has been observed that over half of all software development projects are considered a failure with respect to their cost and schedule [1, 4]. In order to mature the development process in terms of cost, effort and time, an extensive effort is now being made by many of the software industries [1]. In the past decade much effort has been devoted to the application of risk management techniques in the field of software development. Researchers and practitioners have focused on the identification, analysis, and mitigation of the dangers that prevent projects from fulfilling requirements on time and under budget.

Software plays an important role more or less in each and every aspect of human life. But, at the same movement, it has been the most troubling technology of the 20th century [5]. Generally, it has been observed that most of the software projects have the highest probability of being cancelled or delayed of any known business activity. Software projects often display excessive error densities and low levels of reliability on deployment. Because of its highly unreliable and disastrous nature, it is achieving a very bad public reputation. A careful implementation of risk analysis and mitigation may reduce the probability of major disasters [6].

Risk is the possibility of suffering harm or loss which causes danger. In a technology driven area of study, software project risk management has become relevant

[1]Department of IT, Babasaheb Bhimrao Ambedkar University, Lucknow, UP, India. e-mail : khanraees@yahoo.com
[2]Department of Computer Science, Jamia Millia Islamia, New Delhi, India e-mail : kmfarooki@yahoo.com

over time and indicates the importance and complexity of the issue they are addressing. Not only has the issue of software risk management been researched for more than two decades now, but, as years go by, it appears to have attracted the attention of an increasing number of researchers [7]. It is evident from the literature survey that several aspects of software project risk management have been studied using a variety of approaches. It has been observed that the risk management of software project has been an area of research since 1970's and many of academicians and industry practitioners are producing their views on the same [8]. In 1999, a survey conducted on 34 empirical studies on software risk management published during 1978-1999, reveals that the area is enrich in terms of approaches, ranging from action research and case studies to survey and laboratory experiments [9].

This paper is organized as follows: Section 2 introduces risk management including the Barry Boehm's risk factors in managing software projects. In section 3, a risk management process has been introduced. Section 4 proposes a framework to integrate software project risk within the development life cycle. Contextual findings and conclusions are made in section 5.

2. Risk Management

Software intensive development projects still fail to be delivered a quality end product on time and within budget. One area of concentration in software project management that has developed to solve these problems is Risk Management, which attempts to assess and then control the risks that precipitate them [10]. Risk is the possibility of suffering loss which describes the impact to the project in the form of diminished quality of the end product, increased costs, delayed completion, or outright project failure. Risk is uncertainty or lack of complete knowledge of the set of all possible future events. It can be classified as either favorable or unfavorable future events [11].

Understanding the internal and external project influences that can cause project failure is called risk management. It consists of risk assessment and risk control. Risk assessment addresses whether the software intended to use is good enough for the task at hand. Once the project plan is built, a risk analysis will be performed on it. The result of the initial risk analysis is a risk plan that is being reviewed regularly and adjusted accordingly. The main purpose of risk management is to identify and handle the uncommon causes of project variation.

Dangers in the new and emerging field of software engineering must often be learned without the benefit of lifelong exposure. An approach involving studying the experiences of successful project managers as well as keeping up with the leading practitioners and researches in the area is required. The major objectives of software risk management includes identifying, addressing, and eliminating potential elements of risk before they become either threats to successful software operation or major sources of software rework. Barry W. Boehm, in his article Software Risk Management: Principles and Practices listed out following factors associated to risk in managing software projects [12].

- Personnel Shortfalls
- Unrealistic schedules and budgets
- Developing the wrong functions and properties
- Developing the wrong user interface
- Gold-plating
- Continuing stream of requirements changes

- Shortfalls in externally furnished components
- Shortfalls in externally performed tasks
- Real-time performance shortfalls
- Straining computer-science capabilities

3. Risk Management Process

In spite of plenty of work in the area of software project management, it has been commonly accepted by the researchers and industry professionals that the software development projects still fail to deliver acceptable systems on time and within budget. It has also been revealed from the survey report that pro-active planning and considering risk factors may assist in reducing the occurrence of failure. It is better to chock out an accurate and efficient plan and to identify the risks within the development process in order to timely predict and mitigate the flaws, rather than waiting for problems to occur and then trying to react. Risk management has been proposed as a solution to provide insight into potential problem areas and to identify, address, and eliminate them before they derail the project [10].

The objectives of software risk management are to identify, address, and eliminate software risk items before they become threats to success. A good project manager is supposed to be a good manager of risk. In order to achieve success in software development strategy, it is recommended that the development process should incorporate risk management as part of project management [6]. Figure 1 depicts a process of risk management consisting of six phases including identification, analysis, prioritization, planning, resolving and monitoring. Following section describes each phase and its activities in detail.

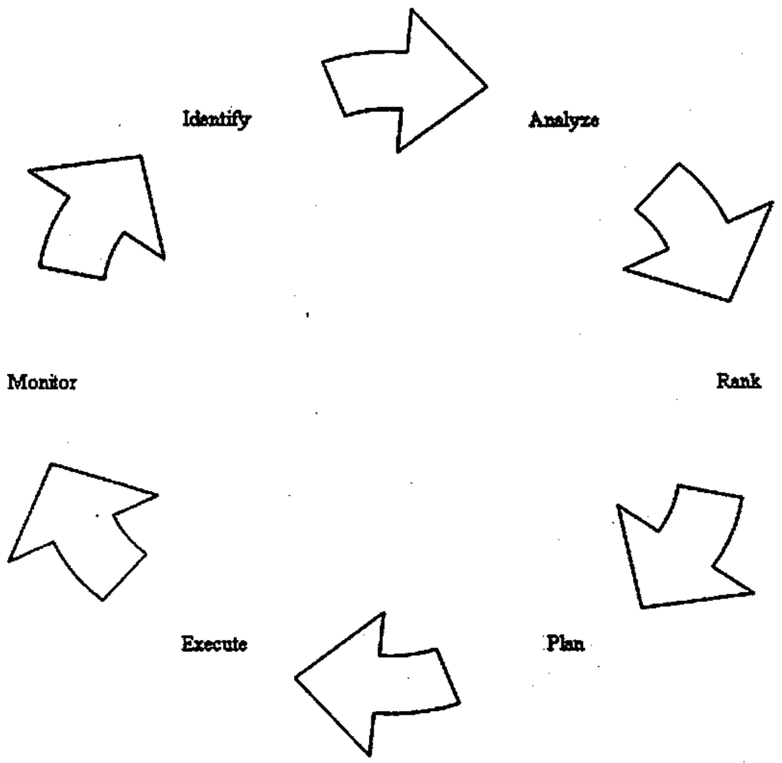

Figure 1: Software Project Risk Management Process

3.1 Identify

Risk may be defined in terms of risk exposures, risk impacts or risk factors. Identifying risk is one of the biggest challenges in managing software project. It forms the basis for managing risks in order to deliver reliable and risk free system in time and budget. The purpose of identifying risk is to consider the risks before they actually become problematic and to incorporate this information into the project management process. It has been observed that various risk items have different impacts, and therefore, specific project risk items are required to be identified in advance for the successful completion of software project.

3.2 Analyze

After identifying risks, it is to be critically examined by the experts in order to propose a solution to overcome the identified problem. Analyzing risk includes determining the probability of loss to be incurred. The purpose of analysis is to convert the data into decision-making information. The cost associated with each identified risk items is examined. This will help managers to reduce the budget overhead. In time identification and analysis of software project risks assists software development team to control and manage software under development.

3.3 Rank

No doubt, identification and analysis of risk may produce a long list of associated items with varying impacts on software project. Therefore, there is a need to prioritize the identified risk items in order to mitigate them accurately and in time. Risk prioritization produces a ranked order of risks identified and analyzed. As a result, the risk will be handled according to their ranks.

3.4 Plan

It is well accepted fact that planning is a key to success. Therefore, an appropriate risk management plan is to be chocked out in order to address the identified risks according to their priorities. The plan outlines the way to mitigate the risk items to control the overall risk of the software under development. This also tells how to integrate individual risk plans into overall project management plan.

3.5 Execute

After planning, the very next step is to execute the set plan. Resolution of risks executes the risk management plan. As a result the actions or activities implemented either eliminate or resolve the risk involved with the particular risk item.

3.6 Monitor

This is the last and very important step in risk management process. After the risk is identified, analyzed, prioritized, planned, and resolved, a corrective action will be taken to monitor the progress of the project in the light of risk management. It is the most challenging task of the process of managing risk of software project, which requires a skilled manpower to accurately monitor the activities performed during risk mitigation.

4. Software Project Risk Within The Development Life Cycle

Acquisition, development, and deployment of software projects continue to suffer large cost overruns, schedule delays, and poor technical performance. Ample literature exists on the process of risk assessment and management. The majority of these literatures, however, are devoted to theories and methodologies that have not been subjected to the ultimate test of practice. There is a need for comprehensive framework to be developed, deployed and tested for the release of risk free software project. It has been observed that cost generally increases with the phases of software development life cycle for the impact of errors shown in figure 2. Therefore, it is well justified to make an effort to fix the bugs and errors early in the development life cycle in order to produce safe software.

Time is one of the critical factors contributing to risk. Therefore, an early act before a source of risk evolves into a major crisis is highly recommended. It has been observed that being reactive in risk mitigation and control rather than proactive in risk prevention and control is at the heart of good risk management. Software project risk assessment is not an independent activity to be carried out at a specific phase or time. Rather, it should be integrated within the development life cycle in order to enable engineers, managers, and other decision makers to identify, sufficiently early, the risks associated with software acquisition, development, integration, and deployment. In order to prevent project runaways, meet deadlines, stay within the project's budget, and simultaneously maintain the product's high quality standards, it is essential to timely identify, analyze and mitigate risks associated with each and every phase of software development life cycle. This will help the management to adapt appropriate mitigation strategies on a timely basis. Integrating risk within the software development life cycle provides a structured process, supported by methods and tools, for identifying, analyzing, and mitigating the uncertainties encountered in a specific software engineering effort.

4.1 Safe Requirement [6]

Requirement specification phase collects complete, unambiguous and understandable requirements from the end-users, and quickly stabilizes the requirements specified. The major risks associated with this phase are that the wrong software will be developed, the software will not be completed on schedule, and the requirements will not be testable. In order to minimize the risk, customers requirements needs to explicitly, accurately and completely stated. It is generally accepted that poorly written and rapidly changing requirements are a principal source of project risk, which may lead to project failure. Depending solely on a customer's high level requirements could lead to ambiguity and errors in later life-cycle activities.

There are various attributes associated with the requirement specification. Some of the pertinent attributes have been identified and defined in table 1 [6]. Table 2 defines the metrics for the identified requirement attributes.

No doubt, later in the life cycle changes are made to requirements, the more resources needed to implement them. It is also observed that late requirement changes may cause a ripple effect, causing additional changes in associated areas. The earlier in the life cycle the requirements stabilize, the lower the risk. It is commonly accepted fact that heavy cost and high risks are associated with the changes made late in the life cycle. Therefore, it is recommended to measure requirements attributes and risk throughout the life cyle.

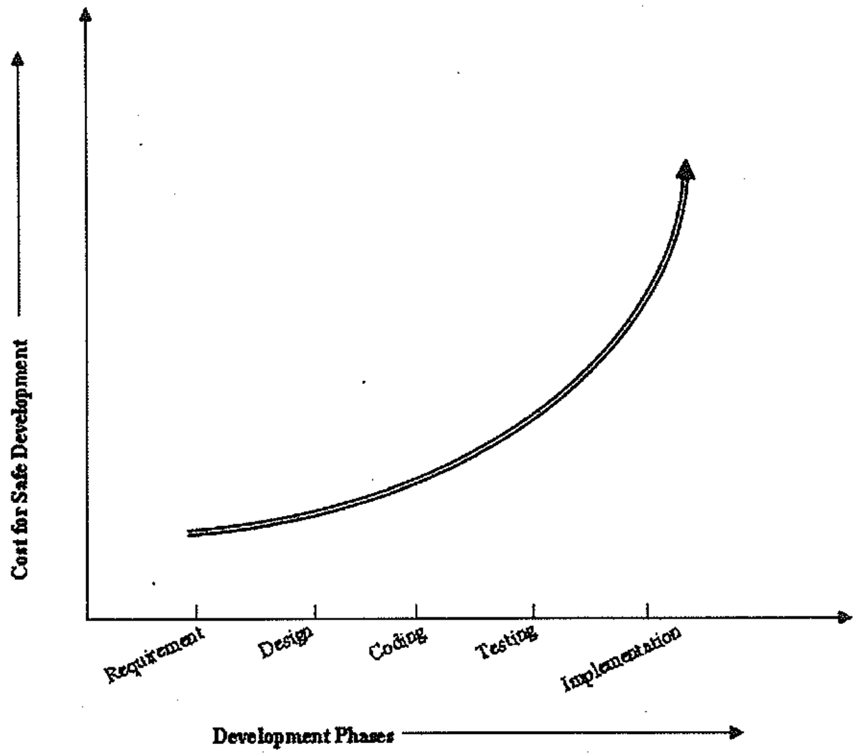

Figure 2: Cost For Safe Development With Phases

Table 1: Software Requirement Specification Attributes

Attributes	Definition
Ambiguity	It is the property of being ambiguous.
Completeness	Code posses the characteristics completeness to the extent that all its parts are present and each part is fully developed.
Understandability	Code posses the characteristics understandability to the extent that its purpose is clear.
Volatility	It is the measure of the state of instability.
Traceability	It describes the ability of the mechanism and its design process to provide links between requirement specification, design, code and test [13].

Table 2: Requirement Metrics

Attributes	Metrics	Definition
Ambiguity	WPM	This metrics identifies ambiguous, optional and weak phases in the requirement specification.
Completeness	TBD, TBA, TBS	This metrics counts number of items not yet specified. It will measure the items to be added, to be determined, and to be supplied.
Understandability	RDM	This metrics measures the readability of requirement documents.
Volatility	VLM	It counts the ratio of number of requirements changed to the total number of requirements.
Traceability	TRM	Traceability metrics measures the percentage of trace up and down.

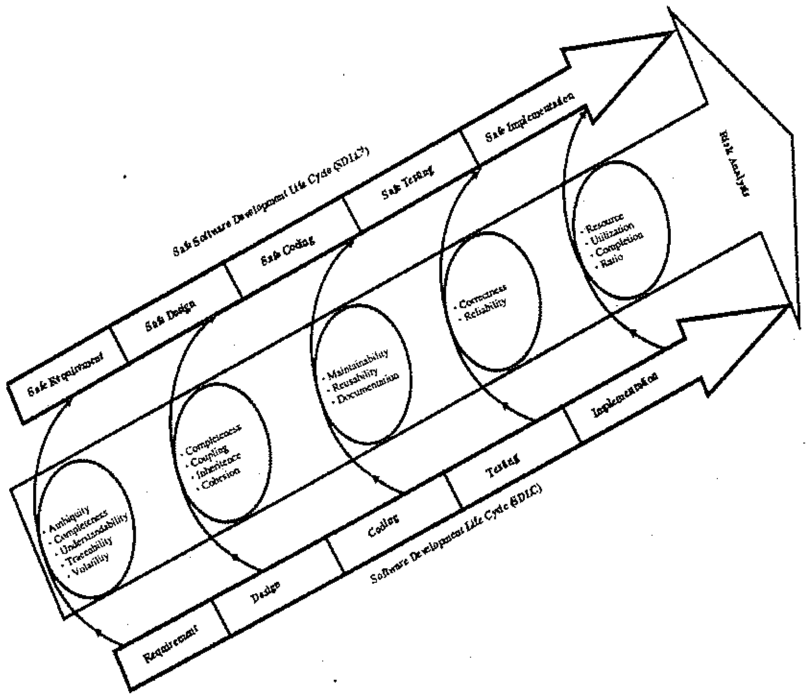

Figure 3: Integrating Risk Within The Development Life Cycle

4.2 Safe Design

Software design is the skeleton of any software, which serves well as a communication medium between the designer and the user on the one end, and act as a basis for implementation on the other end. Design is an important stage spanning the whole software lifecycle, not only for software development but also for re-developing legacy systems [14]. It is concerned with accurately mapping the requirements from the analysis stage to logical models for implementation. The risk assessment at software design heavily affects the risk associated with the final products. Controlling and improving software design risks have been one of the important issues in software project risk management. Because of unavailability of any standard formats for complete and detailed design, metrics for this phase are often ignored or omitted from risk evaluation.

Generally, design of software project is depicted by control flow, data flow, or object-oriented/functional structures. Since the design phase is the transition between requirements and code, most of the attributes and metrics identified for design overlap these two phases. Table 3 shows design attributes and their definition. Table 4 defines the metrics for the design attributes.

Table 3: Design Attributes

Attributes	Definition
Completeness	Completeness defines the Items left to be specified.
Coupling	Defined as the interdependency of an object on other objects in a design.
Inheritance	It is a measure of the 'is-a' relationship between classes.
Cohesion	It is the high degree of internal relatedness of elements

Table 4: Design Metrics

Attributes	Metrics	Definition
Completeness	CMP	This metrics measures the number of modules, not at the lowest level, whose structure is not specified [6].
Coupling	COM	This metric count of the different number of classes that a class is directly related to.
Inheritance	INH	This metric count of the number of class hierarchies in the design.
Cohesion	LCOM	This metric computes the relatedness among methods of a class based upon the parameter list of the methods [computed as LCOM, 1993 Li and Henry version]

4.3 Safe Coding

The ultimate objective of software professionals is to write code and document it in a proper manner by keeping the projects requirement in mind. Writing code for a module is entirely dependent on the design details from the design phase. This phase will form the basis for testing the module for which code has been written. Therefore, in order to perform an optimal test and to write test cases, it is highly demanded to develop an accurate, reliable, understandable, maintainable and reusable code. Moreover, reusability of code also depends on how the code has been written. Coding also decides the maintenance cost of the developed project. These two major attributes of code contributes to risks in this phase. Table 5 shows the attributes in this phase, and table 6 represents the metrics along with their definition.

Table 5: Code Attributes

Attributes	Definition
Maintainability	This is the ease of finding and fixing an error.
Reusability	This defines the feasibility of reuse of software
Documentation	Documentation defines adequacy and usability of internal and external documentation.

Table 6: Code Metrics

Attributes	Metrics	Definition
Maintainability	MBL	This metrics counts the number of linearly independent test paths.
Reusability	RUM	It counts the number of calls to and from the modules
Documentation	DOM	This metrics counts the percentage of comments in code and gives measures to readability of documents.

No doubt, a module with higher complexities is more difficult to understand than a module with lower complexity, as the complexity has a direct impact on maintainability and reusability, in general. Therefore, risk is associated with the module to be maintainable and reusable. Modules that have a high complexity for a few numbers of executable statements may be of highest risk. These modules are tending to be very difficult to understand, increasing the difficulty of maintenance and decreasing the possibility of reuse [6].

4.4 Safe Testing

Software development processes typically focus on avoiding errors, detecting and correcting software faults that do occur, and predicting reliability after development. It is believed that software industry is at a risk for a disaster of some kind, a disaster in which the blame will clearly lay on the software. Many computer systems are used in critical applications such as spacecraft and defense systems. When lives and fortunes depend on software, software quality and its verification demand increased attention. Increasing emphasis placed on high quality and customer satisfaction of software calls for rethinking on the objectives and management of testing. Software testing has an obvious role in finding bugs, and less obvious role in evaluating reliability. Test and evaluation methods and tools, in themselves, do not guarantee effective testing and ensure high quality of software. Testing is often considered an expensive and uncontrollable process, which takes too much time, costs more than planned, and offers insufficient insight of the quality of the test process. Testing is a process of planning, preparing, executing and analyzing the difference between the actual status and the required status [15].

Table 7 : Test Attributes

Attributes	Definition
Correctness	Correctness of a piece of code is defined as the likely number of errors remaining in that piece of code.
Reliability	Reliability of a module is defined as the likely number of highly critical errors remaining in that module.

An effective testing locates and repairs faults in the software, and also identifies error-prone modules in the software under development. The associated risks are to schedule the optimal test and produce reliable software. A piece of code with more than the average number of errors is supposed to be highly risky, and needs to be investigated. Several attributes have been identified by the various researchers and practitioners for testing a piece of code, some of them are discussed in table 7. Table 8 shows the metrics for each identified attribute in testing phase [6].

Table 8 : Test Metrics

Attributes	Metrics	Definition
Correctness	COR	It counts the percentage of errors within a program that must be removed.
Reliability	REB	It estimates the number of errors left in the software that are of high criticality.

4.5 Safe Implementation

This is not a specific phase, but starts with the requirement specification and continues till release of software project. The major objective of this phase is to maximize the utilization and effectiveness of resources in time and within project budget in order to carry out the various activities during development [6]. Pertinent attributes identified to accomplish the goals of implementation are described in table 9, and their corresponding metrics are defined in table 10.

Appropriateness of tasks assigned in a specific phase will form the basis for risk at that particular phase. There may be chance that the tasks for which resources are being utilized during development do not match with the expected or planned activity, leading to high risk.

Table 9 : Implementation Attributes

Attributes	Definition
Resource Utilization	This defines the utilization of resources to carryout various activities corresponding to different phases of software development life cycle.
Completion Rate	This defines the progress rate for completion of any activity during development.

Table 10 : Implementation Metrics

Attributes	Metrics	Definition
Resource Utilization	REM	It measures the resources utilized for completion of an activity.
Completion Rate	COR	This counts the rate of completion of assigned activity.

5. Conclusion

Software risk management is the formal process in which risk factors are systematically identified, assessed, and mitigated. The determination of the risk in a project either due to external or internal causes is a major part of project management. Risk analysis is a good general-purpose yardstick by which security effectiveness can be judged. It has been observed that roughly 50 percent of security problems are the result of design flaws. Therefore, performing a risk analysis at the design level is an important part of a solid software security program. Taking the trouble to apply risk-analysis methods at the design level for any application often yields valuable, business- relevant results. The process of risk analysis is continuous and applies to many different levels, at once identifying system-level vulnerabilities, assigning probability and impact, and determining reasonable mitigation strategies.

Design, Coding and installation phases have the highest project execution risk reduction potential. It has been observed that the risk continues to be reduced and the value of the project investment smoothly increases.

References

[1] D. Merrill, *"Software Development Project Managers with a Software Project Simulator,"* Master of Science Thesis Proposal, Department of Computer Science and Engineering Arizona State University Training February 4, 1996.

[2] B. W. Boehm, *"Software Engineering Economics"*, Prentice-Hall, Englewood Cliffs, New Jersey, 1981.

[3] D. H. Kitson and S. Masters, *"An Analysis of SEI Software Process Results 1987-1991."*, Proceedings of the Fifteenth International Conference on Software Engineering, PP. 68-77, 1993.

[4] C. Jones, *"Applied Software Measurement: Assuring Productivity and Quality"*, McGraw-Hill, New York, 1991.

[5] Jones, Capers, *"Minimizing the Risks of Software"*, May 1998.

[6] A. D. Buttigieg, *"Risk Management in a Software Development Life Cycle"*, Web reference. Available at: http://www.cis.um.edu.mt/~abut/#Section%202

[7] H. Barki, S. Rivard, and J. Talbot, *"An Integrative Contingency Model of Software Project Risk Management"*, Journal of Management Information Systems / Spring Vol. 17, No. 4, PP. 37-69, 2001.

[8] M. Keil, P. E. Cule, K. Lyytinen and R. C. Schmidt, *"A framework for identifying software project risks"*, Communications of the ACM, 41, 11, 76-83, 1998.

[9] J. Ropponen, *"Software Risk Management: Foundation"*, Principles and Empirical Findings. Jyväskylä: Jyväskylä University Printing House, 1999.

[10] R. Bechtold & P. McNeece, *"Managing Risk With Metrics: A Term Paper for the MJY Team, Software Risk Management WWW SITE"*, TP-PM Final, 21 April 1997.

[11] Don Shafer, *"Risk Software Risk: Why must we keep learning from experience? Dynamic Positioning Conference"*, Athens Group, Inc., Houston, Texas, USA, September 28-30, 2004. Available at: www.athensgroup.com

[12] *"Introduction to Software Risk & Risk Management"*, April 24, 1997, Web Referencehttp://www.baz.com/kjordan/swse625/intro.html

[13] S. Martin, *"Software Security Evaluation Based on a Top-Down Mc Call-Like Approach"*, IEEE 1988, PP. 414-418.

[14] G. Peterson, *"Collaboration in a secure development process, part 1, Information Security bulletin"*, 9:165-172, 2004. Available at: http://www.arctecgroupnet/ISB090GP.pdf

[15] K. Mustafa & R. A. Khan, *"Software Testing: Concepts and Practices"*, Narosa Publication, 2007.

Author's Biography

Dr. R. A. Khan is currently working as a Reader in the Department of Information Technology, Babasaheb Bhimrao Ambedkar University (A Central University), Lucknow, UP. His area of interest is Software Security, Software Quality and Software Testing. He has authored two books on software quality and software testing.

Dr. K. Mustafa is currently working as a Reader in the Department of Computer Science, Jamia Millia Islamia New Delhi-India. Dr. Mustafa has published several papers and articles in Internationals and National Journals. He is the author of books 'Software Quality: Concepts and Practices" and Software Testing: Concepts and Practices'.