A Study on Software Risk Management Strategies and Mapping with SDLC

Bibhash Roy, Ranjan Dasgupta and Nabendu Chaki

Abstract In recent years, despite several risk management models proposed by different researchers, software projects still have a high degree of failures. Improper risk assessment during software development was the major reason behind these unsuccessful projects as risk analysis was done on overall projects. This work attempts in identifying key risk factors and risk types for each of the development phases of SDLC, which would help in identifying the risks at a much early stage of development.

Keywords Risk management · Risk models · SDLC · Technical risk

1 Introduction

Software project management is crucial for development, services and maintenance of software products. Management of diverse activities during software engineering process needs to be handled carefully for any software project. One of the most important yet often overlooked aspects in the complete process is risk and its management [1]. A risk may be considered as a probabilistic term that has the potential to affect the overall project in a negative way. Failure of the projects, especially IT projects, is often due to these unwanted and rather less explored

B. Roy (✉)
Tripura Institute of Technology, Tripura, India
e-mail: bibhashroy10@yahoo.co.in

R. Dasgupta
National Institute of Technical Teachers' Training and Research, Kolkata, India
e-mail: ranjandasgupta@ieee.org

N. Chaki
University of Calcutta, Kolkata, India
e-mail: nabendu@ieee.org

R. Chaki et al. (eds.), *Advanced Computing and Systems for Security*,
Advances in Intelligent Systems and Computing 396,
DOI 10.1007/978-81-322-2653-6_9

threats or issues [2, 3]. Some works during the last two decades on risk management strategies have emerged as survival component for software projects [4, 5]. However, it is important to identify the possible risks in all the stages of the software development process so that proper mitigation strategy can be adopted at the appropriate level to reduce the possible financial and temporal loss.

Any software project may face different categories of risks, either internal or external, in its engineering process. There are different kinds of risks encountered in different phases of software development life cycle (SDLC) and these risks can be classified as technical risks and nontechnical risks. The internal risks are due to the factors within the organization and external risks come from outside the organization and are difficult to control. Software risks can be grouped into project risks, process risks, and product risks. Risks for software project management have been classified as Known risk, Unknown risk, Predictable risk, Unpredictable risk [3]. Another approach used in [6] identifies risks as Technical risk, Management risk, Financial risk, Contractual and legal risk, and Personal risk. On the other hand in [7], it was observed that risks are identified as Business risk, Commercial risk, Economical risk, Project risk, Product risk.

In the last few decades the researchers all over the globe had done lots of works and various risk models had been proposed and applied at different cases with various levels of success. All such models have lots of merits; however, no single model can be applied to all cases. Moreover, for a complex and big projects looking the risk analysis from the top level might not be appropriate as the risks occurred in the lower level might not be identifiable if the tool for identification is not applied at that level where the actual risk might occur. In this paper, risk management models are reviewed in such a way that the models can be mapped with the different stages of the life cycle of a software development process. For this purpose, major objectives and capacities of various risk management models had been studied (Sect. 2). All steps of activities of a software development process had also been studied with an eye to identify the types of risks that may occur at each level (Sect. 3). In Sect. 4, mapping has been done from the outcomes of Sects. 2 and 3 so that insight knowledge can be extracted from the final result to design the mitigation strategy.

2 Risk Management Models

Risk management is a set of activities required to manage risk. Organizations who apply risk management methods and techniques have greater control over the projects [8]. The risk management methods that are being proposed till date follow mainly two characteristics—probability and impact. Most of the proposed methods are static in nature that performs qualitative and quantitative analysis to assess and control the risks. Most of the methods divide the risk management into some basic processes like *risk identification, risk analysis, risk planning or mitigation, risk monitoring and control,* etc. [3, 9]. It is very much essential to identify the causes as

well as effects of the risks for a comprehensive analysis of risks and many of the methods perform this but concentrate on a single risk.

One of the oldest as well as mostly discussed risk management models is Barry Boehm's (1991) BOEHM Model [1–3, 10] that emphasizes on the concept of 'risk exposure' and can be applied to almost any software-related project. BOEHM divides the risk management into two primary steps—*Risk assessments (risk identification, risk analysis, and risk prioritization) and risk control (risk management planning, risk resolution, and risk monitoring)*. Boehm proposed an effective risk control strategy where each risk is thoroughly tracked with the aim that this risk either can be eliminated or its effect into the project in terms of cost or time can be reduced to certain extent. Boehm has discussed different tools like checklists, cost models, cost–benefit analysis, etc., in each step, however, no standard project metrics was used as a tool for risk management. This method can be applied in all the phases of SDLC as risk analyzer. However, new risks that are occurred during the development process were not taken into consideration.

Software Engineering Institute in 1993 proposed SEI-SRE [2, 3, 11, 12] (Software Engineering Institute, Software Risk Evaluation) model that provides a framework for risk evaluation to overcome project failures. This model concentrates on practical aspects of a project that makes it eligible to be applied successfully in IT projects as a decision-making tool. However, there is no scope for modification in the template-based design and sometimes inconclusive outcomes might occur as it mostly relies on the experiences of project personnel. Unlike Boehm, it has proper method to measure and evaluate the effectiveness of risk treatment; however, fails to perform review of risk information periodically as performed by Boehm model.

Ronald P Higuera et al. in 1994 proposed Team Risk Management (TRM) Process Set [5, 12, 13] that emphasizes on team structures and its activities for managing risk in each phase of SDLC involving all individual connected to the project to ensure continuous risk management throughout the project development. This approach follows continuous risk evaluation process where new risks are considered and mitigated or resolved risks are automatically removed from the threat list and thereafter updated risk status is communicated to all the individuals connected in the project. Like TRM, Agle et al. (2003) also proposed [14, 15] a risk handling mechanism related to team structure in multi-team environment.

Another widely recognized method developed by Jyrki Kontio et al. in 1996 is known as RISKIT [2, 3, 12, 16]. This risk management cycle performs *risk identification, risk analysis, risk monitoring,* and performs qualitative analysis to prioritize according to probability and impact. It uses a graphical method called Risk Analysis Graph (RAG) to monitor risk scenario development and the concept of utility loss is used to assess the impact of risk. This model may be applied mainly in large organizations, IT projects, and Nokia Telecommunications is one of its major users. However, it is very much flexible to be implemented in different categories of projects or areas such as business planning, marketing, and technology-related fields. Due to the absence of correlation between risk estimation and risk metrics, this method suffers inaccurate prediction possibilities of potential risks.

In the year 1997 two different models were proposed: Software Engineering Risk Understanding and Management (SERUM) [2, 3, 17] by D. Greer and Risk Maturity Model (RMM) [18] by David A. Hillson where SERUM performs both *implicit risk and explicit risk* management and RMM measures the maturity level of an organization. Though SERUM is typically designed for software projects, it suffers time management problem in analyzing risks. Unlike other maturity model such as CMM, EFQM, etc., RMM is completely based on risk management. It performs different risk management approaches of an organization to assess the maturity level in managing risks.

Software Risk Assessment Model (SRAM) [19] and SoftRisk [5, 12, 20] are the two different risk management approaches that came out during the year 2000 where SRAM uses a set of comprehensive questionnaire for different critical risk elements to prioritize risks and SoftRisk generates a graphical tool helping the managers to perform risk control mechanism. Like TRM model, SoftRisk approach also considers the new risks in its continuous risk management process.

In 2007, William G. Snekir et al. proposed Enterprise Risk Management (ERM) method that performs risk assessment with the help of graphical decision tree and quantitative analysis.

Armestrome and Adens (2008) [15, 21] gave a very decent idea of identifying most prominent area of exposure to risk to characterize those areas into different risk factors and to prioritize them to perform risk management strategies. On the contrary, Software Risk Assessment and Estimation Model (SRAEM) [3, 4, 22] having weak risk identification strategies estimates the sources of risks from different paradigms and compute prioritization and ranking by MCRSRM (Mission Critical Requirements Stability Risk Metrics) followed by quantitative assessment. Unlike SRAEM, Analyzer for Module Operational Risk (ARMOR) [23] automatically identifies the operational risks of software program modules. Being a module-based approach this model can perform risk management in every stage of software development and like SoftRisk model, can perform various types of data analysis. A modified approach of SRAEM is Software Risk Assessment and Evaluation Process using model-based approach (SRAEP) [3, 4, 24] that aims at risk assessment and risk prioritization. This model uses SFTA (software fault tree approach) to identify and analyzes the risk and uses RRL (risk reduction leverage) for risk measurement.

Hoodat and Rashidi (2009) [15, 25] have proposed different classifications of risk based on indexing of different risk factors followed by calculating their impact on software project. Different categories of risk identified by them are (a) *Internal risk and external risk.* (b) *Process risk, product risk, and project risk.* (c) *Performance risk, cost risk, and scheduling risk.* (d) *Requirement risk, cost risk, scheduling risk, quality risk, and business risk*. A similar risk management activity performed by Software Project Risk Management model (SPRMQ) (2011) [7] requires experience of project manager to manage software product risk. However, unlike other models it does not consider the external risk. Another approach as proposed by DANNY in 2006 [15, 26] reduces operational risk by performing risk

classification and their qualitative analysis with an aim to save resources with a consideration for small-sized projects or available funded projects.

Suebkuna and Ramingwong (2011) refined the approach of Project Oriented Risk Management Model (PRORISK) [27] with a proposal to link project management and risk management having two phases of risk management: risk assessment and risk control. It uses a risk database to record risk control related information such as its impact, probability, type, mitigation strategy, etc.

Shahzad et al. (2011) proposed risk "handling and avoidance mechanism" in Risk Identification, Management and Avoidance Model (RIMAM) [3, 28]. This model provides a stepwise procedure of risk handling methodology that enables the development team to handle risks locally. This model is suitable to be applied in small and medium scale of software with tight budgetary and short-time period.

Project Risk Network Model (PRM) [6, 12] as proposed by Linda Westfall (2011) is one of the well-known methods detailing a Decision Support System framework consisting of five major steps: risk identification, risk assessment, risk analysis, risk response planning, and risk monitoring and control enabling project managers in choosing a set of risk mitigation action with minimum loss. The drawback of this model is that it uses the classical method for risk identification and evaluation during its initial phases with an assumption that risks are independent, whereas there are lot of interactions and influences between different risks in the project development.

Software Engineering Risk Index Management (SERIM) (2014) [2, 3, 29] is one of such risk management models that is typically designed for software project development focusing mainly on the assessment of risk factors (*Organization, estimation, development, methodologies, tools, risk, culture, usability, correctness, reliability, and personnel*) with periodic measurements on high priority risk areas throughout software development stages. Though this method lags in providing explicit guidelines to identify risks that may involve in the project, medium-sized organization may consider it instead of expensive methods.

Most recently Loutchkina et al. (2014) proposed System Integration Technical Risks' Assessment Model (SITRAM) [12] that integrated Bayesian Belief Networks (BBN) and Parametric models (PM) providing statistical information for improving risk management during large software development.

In this section, we presented observations on some of the well-known risk management models. These models cover really an wide and diverse range of objective right from identification of risks to risk mitigation, risk assessment, controlling of risks, prioritization of risks, risk probability computation, and even proposing a four level of risk maturity model. Some of the existing risk models are good specifically for either small or large organizations. On the other hand there exist risk models that especially deals with risks associated with software projects only. Many of these have been implemented using suitable tools. However, in the current state scenario only a few models exist that considers risks associated with different stages of SDLC. The study further reveals that such models, in use, lack

Table 1 Observations of different risk models

SN	Methods/models/proposed	Observations
1	BOEHM [10]	Does not handle generic risk; works on risk analysis paradigm principle
2	SEI-SRE [11]	Generates a template-based design that results in inconclusive outcomes due to less scope for modification
3	RISKIT [16]	Does not collaborate risk estimation and risk metrics, thus reducing the prediction possibilities of potential risks
4	SERUM [17]	As it performs a continuous evaluation of risks, hence time management holds the key role as risk element in the project
5	SERIM [29]	Good for small organizations; handles multiple projects for analyzing software risks; lacks explicit guidelines on using information to identify possible risks in the project
6	SRAM [19]	Risk ranking is done by AHP and entropy method. It does not handle marketing risk
7	Agle et al. [14]	Handles team structure; does not consider funding and resources
8	Danny [26]	Performs classification of risk by quantitative analysis; aims at saving resources
9	Armestrong [21]	Identifies the risk exposure areas and prioritizes them in respect to business context
10	Rashidi [25]	Perform risk classification and risk indexing
11	SRAEM [22]	Risk prioritization and ranking is computed by MCRSRM, decision through quantitative assessment; model focuses on external risks related to the requirement analysis
12	SRAEP [24]	Model uses SFTA to identify and analyses the risk and RRL for risk measurement; follows models based approach
13	SPRMQ [7]	Well suited for handling the product risk; does not consider external risks such as marketing risk, organizational risk, etc.; uses avoidance, minimization, and contingencies strategies
14	RIMAM [28]	Works on the principle of "*handling and avoidance mechanism*"; some of the risks can be handled locally
15	TRM [13]	Follows all the steps of SEI; handles new risks and risk status are communicated to all individuals
16	SoftRisk [20]	Documents all types of risks; performs qualitative and quantitative analyses; Consider new risks in an iterative process
17	ARMOR [23]	Identifies source of risk and suggests solution to reduce risk levels; uses regression analysis to validate generated risk model
18	RAT [41]	Performs hybrid assessment of risks in five phases; risks are ranked based on ranking matrix

(continued)

Table 1 (continued)

SN	Methods/models/proposed	Observations
19	ERM [42]	Evaluates level of an organization to propose risk assessment tool using graphical decision trees and quantitative analysis
20	PRORISK [27]	Links project and risk management toward developing a risk database; handles six types of risks for software projects
21	RMM [18]	Provides the benchmark to an organization to assess its maturity level in terms of project risk management
22	PRM [6]	Works on the assumption that risks are independent which may lead to incorrect risk assessment
23	SITRAM [12]	Provides statistical information in decision-making of risk management using Bayesian belief networks and Parametric models

completeness from one or more perspectives. Based on the above discussion, we have summarized the key features of these models in Table 1. However, other research works [15, 30–40] deal with various special issues like Risk management in requirement engineering, Risk-based testing, Project Risk dependencies, etc., and not included in Table 1.

Based on above observations of risk management models under discussion it can be summarized as indicated in Table 2 that different models are strong enough to be applied in different application domains.

Risk management models that are discussed in this section involve different participants in its risk management process and are summarized in Table 3.

Among the risk management models considered in our discussion, only few models emphasized on the importance of project size in its risk management strategies. It is general consideration that if a risk management model can handle a large-sized project then it will surely be able to handle medium-and small-sized project too. The medium- and small-sized projects face mainly two constraints: very tight schedule and a limited budget [34]. These constraints do not permit to go for a risk management strategy that may incur more costs and time [15]. Rather than going for those complex and costly risk management models, a simple and effective model would be used for relatively smaller projects. It has been found that there are monitory losses in medium and smaller projects due to improper handling or not considering the risks related to the projects [34]. During last decade the concentration has been given on the risk management strategies for small and medium projects [2, 3, 5]. Different researchers have proposed risk models for small and medium software projects [15]; however, proper and detailed description of management process would require some standards. Risk management models under discussion may be compared on the basis of effectiveness on budget, time, and size of the project and is tabulated in Table 4.

Table 2 Risk models categorization based on their application domain

SN	Methods/models/proposed by	Application domain
1	BOEHM [10], SEI-SRE [11], SPRMQ [7], PRORISK [27], ARMOR [23], RISKIT [16]	Software/IT projects
2	TRM [13], Hoodat_Rashidi [25]	Can be used as team risk management tool in software development process
3	SoftRisk [20], PRM [6]	Flexible to be implemented in different categories of projects of any domain
4	RMM [18]	Can be used by any organization to assess its risk maturity level
5	ERM [42]	To assess an organization's level
6	SERIM [29], SRAM [19], RIMAM [28], SRAEM [22], SRAEP [24]	Small- and medium-sized software projects
7	Danny [26]	Small projects with no costs risk involvement
8	Armestrong [21]	Projects with budget constraints
9	RAT [41]	Web application
10	RISKIT [16], SERUM [17]	Existing software's new version release
11	SITRAM [12]	Large and complex software projects

Table 3 Risk models categorization based on participants

SN	Methods/models/proposed by	Participants in the process
1	BOEHM [10], RISKIT [16], SRAEM [22], SRAEP [24], SoftRisk [20], ARMOR [23]	Risk management team
2	PRM [6]	Risk management team, project manager
3	RIMAM [28]	Risk management team, development team
4	RMM [18], SRAM [19]	Risk assessment team
5	Armestrong [21], Hoodat_Rashidi [25]	Project managers
6	SERIM [29], SPRMQ [7], PRORISK [27], Danny [26]	Project managers, risk managers
7	ERM [42], SEI-SRE [11]	(Experienced) risk managers
8	RAT [41]	Risk managers, project development team
9	SERUM [17]	Software development team
10	TRM [13]	Development team, stakeholders, customers, users
11	SITRAM [12]	Risk assessment team

Table 4 Risk models categorization based on projects size, cost, and application

SN	Methods/models/proposed by	Effectiveness		Applicable to projects size
		Budget	Time	
1	BOEHM [10], RISKIT [16], SERUM [17], PRM [6], RMM [18], Hoodat_Rashidi [25], SoftRisk [20], ARMOR [23]	Tight	Scheduled	Large
2	SEI-SRE [11], SPRMQ [7], Armestrong [21]	Tight	Scheduled	Large, medium
3	ERM [42]	Tight	Scheduled	Large, medium, small
4	SERIM [29], SRAM [19]	Tight	Scheduled	Small, medium
5	RIMAM [28]	Tight	Critical and short	Small, medium
6	SRAEM [22], SRAEP [24]	Tight	Critical	Small, medium
7	TRM [13], RAT [41], PRORISK [27]	Tight	Scheduled	Small, medium
8	Danny [26]	Available	Scheduled	Small
9	SITRAM [12]	Available	Scheduled	Large and complex

3 Risks at Each Level of SDLC

Software project development is a systematic process and risks may occur in every stage of this process. Thus it is essential to look the entire risk management mechanism from a different angle. Conventional approaches of study for internal risk, external risk, known risk, unknown risk, etc., are necessary. However, these cannot be considered as sufficient to identify the chances of occurrence, impact of such risk(s) in terms of cost and time, etc., at each level of SDLC. As for example, if a *risk of conflicting requirements* is left unidentified at the requirement specification level as no conflict identification technique (mechanism) has been applied at that level, the risk will naturally be carried forward and in worst case it might been slipped as undetected and carried up to implementation. The risk, when occurred might not only cause some serious business loss, identification, and mitigation of the problem would also be very cumbersome and extremely difficult. On the other hand, if appropriate mechanism be applied at every stages of SDLC, it not only will identify and arrest a lot of risks to propagate further and cause serious damages, maintenance overhead will also be minimized and more robust software can be offered to the users. Moreover, the impact of the risk might be many folds higher and serious, if it is propagated forward and occurred at a later stage, particularly, when the customer's dependency on the product has become higher and inseparable. Loss of confidence, if happens, on the product and services is another issue which neither can be measured and sometime may lead to irreparable damage and cause change in business sentiments.

The necessity of new risk categorization at a more micro level is thus a necessary requirement where risks can be categorized as per their chances of occurrences in the various development stages. As risk management strategy involves both risk assessment and risk mitigation, hence this classification will help in stagewise risk identification and mitigation during software project development. Software project risks may be grouped as per their occurrences in the phases of SDLC, i.e., requirements and planning, designing, coding, application and maintenance, and other related parameters like scheduling, cost, quality, and business are also affected.

As per SEI [1, 4, 5, 11] the following risk factors may be associated to software development.

1. Incorrect resources estimation
2. User/customer uncertainty
3. Ambiguous requirements
4. Improper design risk
5. Development system and risk with development system
6. Inadequate management process
7. Improper work environment

Unlike other generic project risks, technical risks are matter of concern for software and IT projects and generally lead to failure of functionality and performance [12]. SEI also identifies following technical risks associated with technology related projects

- Lack of strategic framework or conflict over strategy
- Lack of adaptation to technological change
- Supplier/vendor problems
- Poor management of change
- Too much faith in ability of the technology to fix the problems

In the context of SDLC, the various causes or factors have been identified (Tables 5, 6, 7, 8 and 9) as technical risks at different phases against some key factors.

4 Mapping of Models for Different Stages

Different risk management models discussed so far were performed considering overall scenario of the project. Whereas risks in software projects may occur in any of the phases of SDLC and depends on separate strategies adopted for individual phases of SDLC. Software risks have been broadly grouped in literature [6] as nontechnical risk (project and business) or as technical risk. However, risks related to software project development also depend on factors like product engineering, development environment, and program constraint. Again, there are scheduling risks and quality risks that come under planning phase. In addition to software

Table 5 Technical risks at requirements analysis and planning phase

Key factors	Overview of key factors	Technical risks
Stability	Risks due to instability in requirement	Continuous changing requirements
Completeness	Incomplete or unrealistic planning	Inaccurate sizing of deliverables
		Unrealistic time schedule for individual module development
Clarity	Improper requirement or inadequate analysis	Improper definition of requirements
		Inadequate software project risk analysis
Validity	Less or no knowledge about validity of existing tool, inaccurate estimation	Inaccurate cost estimation
		Inaccurate quality estimating
		Inadequate software policies and standards
		Less knowledge about the availability of resources
		Incorrect estimation of resources
		Inability to estimate the scope for reusability of existing modules
Feasibility	Lack of documentation, reusable resources	Nonavailability of documentation of previous projects
		Lack of reusable requirements
		Lack of reusable documentation
Scalability	Scalability of resource requirements	Inaccurate metrics
		Inadequate assessments

Table 6 Technical risks at design phase

Key factors	Overview of key factor	Technical risks
Functionality	Incorrect function design	Inexperienced software module designer
Difficulty	Inability to design or unavailability reusable design	Inadequate tools and methods for quality assurance
		Inadequate tools and methods for software engineer
		Lack of reusable data
		Lack of reusable design
		Lack of reusable architecture
Interfaces	Less knowledge of interface design	Incorrect interface design
Performance	Incomplete designed module	Error-prone module designed
Testability	Unrealistic design	Inexperienced software module designer
Hardware	Little or no knowledge of hardware constraints	Incorrect hardware simulation or interface

Table 7 Technical risks at coding phase

Key factors	Overview of key factor	Technical risks
Functionality	Non-functioning of development team	Inexperienced technical staffs
		Inexperienced development team
		Malpractices of technical staff
Feasibility	Non-mapping between design and coding	Unrealistic module designed to develop using existing technology
		Difficult project modules integration
Testing	Inaccurate or unrealistic development	Wrong integration of modules developed
Availability	Nonavailability of technology, reusable resources, and experience	Unavailability of advanced technology
		The existing technology is in initial stages
		Less reusable code available in the organization
		Inadequate technical training to the staff
		Lack of reusable human interfaces
		Lack of any specialization
		Poor technology investment
		Slow technology transfer
Coding/implementation	Excessive coding and development	Product is complex to implement using existing technology and technical staffs
		Excessive load to development team
		Overschedule due to erroneous coding
		Excessive coding due to lack of standard programming guidelines

Table 8 Technical risks at testing phase

Key factors	Overview of key factor	Technical risks
Environment	Inadequate testing environment	Unable to identify ambiguity in the developed product Lack of reusable test plans, test cases, and test data
Validity	Inexperienced testing team	Inexperienced testing team Inability to identify and to fix problems or errors
Product	Incorrect testing on product	Improper testing strategies
System	Improper developed system	Improper defined objectives to the testing team

Table 9 Technical risks at application and maintenance phase

Key factors	Overview of key factor	Technical risks
Maintainability	Non-flexible, platform-dependent design, high cost	No or less scope for change in the system Platform-dependent application High maintenance costs
Reliability	Non-reliable, non-satisfaction developed system	Unsatisfied customer of the developed product Partially developed product Low productivity
		Low user satisfaction
Safety	Inadequate documentation, tools for maintenance	Overbudget due to corrective maintenance
		Lack of proper help or manuals to the users
		Inadequate tools and methods for technical document
Security	Less or no security on developed system	Lack of sufficient security in the developed product
Human factors	No clarity on functionality of different users	Complex user interface developed
		Nonuser friendly product developed
Specifications	Wrongly specified, underdeveloped system	Inappropriate product developed
		Obsolete product developed
		False productivity claims
Realistic	Unrealistic modification request from customer	Frequent change request due to bugs in the product developed
		Unrealistic change requirements
		Ambiguous improvement targets

engineering phases, risks can further be categorized into performance risks, cost risks, support risks, and schedule risks.

In general, there are many risks in the software engineering process and it is quite difficult to identify all of these. There are still some correlations between management risks versus financial risks, technical risks versus personal risks, business risks versus commercial risks, business risks versus economical risks and between financial risks versus commercial risks.

Since software engineering is a systematic development process, one must follow a proper sequence of steps irrespective of process model in use. None of the above-discussed classifications provides a clear and exhaustive categorization of risks in software projects.

It is, therefore, necessary to propose a new software risk classification where risks are classified based on the stage where they are identified. We propose a novel software risk classification approach where each category of risks follows a single stage of software development.

In this classification, risks identified in a single stage are grouped together. This classification will surely help the project managers to deal with the risks on that particular stage only where it was identified instead of delaying the mitigation till last stage. Following this classification, an early detection and early mitigation strategy can be adopted in order to minimize the loss in software project development.

Risk management strategies considered in most of the risk management models deal with handling generic risks related to the completion of the project and its analysis performed during requirement analysis only. These models highly focus on the requirement analysis and planning phase and try to identify the risks on generic sense. Since software project development is a systematic approach and it has clear distinct phases, a risk management strategy needs to be discussed at macro level where different risk management strategies may be adopted for individual phases of SDLC. Few of the above-discussed models also considered risks related to scheduling and quality during this phase. However, discussions on the risks related to design phase are yet to be considered by these risk models. There are few models [6, 11, 27] that focus on risks related to coding phase, whereas risk considerations related to testing and debugging are being discussed in [11, 25].

Along with generic risks, models as described by RISKIT, Hoodat_Rashidi, Danny also considered risks related to application phase and the risks related to maintenance phase are being discussed by SEI-SRE, RISKIT, PRORISK, ARMOR, SOFTRISK, PRM.

The existing risk management models barely consider software risks according to the phases of SDLC. SEI-SRE model focuses on the phases of SDLC where it emphasizes on the individual factors that may affect individual phases of SDLC.

Though SEI-SRE model is partially capable to deal with the practical aspects of IT projects, there is a limitation in modification of the template-based design. This may lead to inconclusive results for inexperienced user. It deals with risk from product, process, and constraints. It sets a risk baseline for a project. Unlike Boehm's work, SEI-SRE model did not clearly define the responsibilities of the project team.

Consequently, risk status data are not properly communicated between them. SEI-SRE has no method to analyze newly identified risks and it does not perform periodical review of risk treatment. This method only involves project personnel in its risk analysis and thus exclusion of stakeholders may create a possibility of incorrect requirements to the development team that incorporate some more risks which can lead to failure (Table 10).

Hence, a clear mapping is desired so as to understand which risk management model is best suited to follow the phases of SDLC. In our discussions a mapping is drawn in between the existing software risk management models and risks related to phases of SDLC. This mapping indicates a requirement of risk management model that deals risks at each phase individually rather than considering whole project at a time. An early detection of such risks is also essential so as to prepare the mitigation strategy well ahead of occurring of risks in reality.

Table 10 Risk models mapping with phases of SDLC

SN	Methods/models	Purpose	Risk element considered	Risks of which SDLC phases are considered
1	BOEHM [10]	Risk identification, analysis, prioritization, control	Generic risks and project-specific risks	Requirement analysis and planning
2	SoftRisk [20]	Risk identification, assessment, monitoring		Requirement and planning phase, maintenance phase
3	ARMOR [23]	Risk identification, analysis	All program module risks	
4	PRORISK [27], PRM [6]	Risk assessment, risk control	Software-related Generic risks	Requirement phase, coding phase, maintenance phase
5	RMM [18]	Risk assessment	Organizational risks	Not followed
6	ERM [42]	Risk identification, assessment	Generic risks and project-specific risks	
7	RAT [41]	Risk assessment, treatment and monitoring	Project risks of small and medium software	
8	TRM [13]	Risk analysis, mitigation	Team risks	
9	Agle et al. [14]	Risk handling	Risk related to team structure	
10	SEI-SRE [11]	Risk evaluation: Detection, specification, assessment, consolidation, mitigation	Product risks, process risks	Requirement phase, coding phase, testing phase, maintenance phase
11	SRAM [19]	Risk assessment, prioritization	Development risk	Requirement analysis
12	Armestrong [21]	Risk identification	Economic risk, business risk	
13	RISKIT [16]	Risk identification, analysis, monitoring, prioritize as per probability and impact	Generic risk, project risk, technical risk, schedule risk, business risk	Requirement phase, application and maintenance phase
14	H. Rashidi [25]	Risk measurement	Project risk, product risk, schedule risk, cost risk, quality risk, business risk	Planning phase, testing and debugging phase, application phase

(continued)

Table 10 (continued)

SN	Methods/models	Purpose	Risk element considered	Risks of which SDLC phases are considered
15	SERIM [29]	Risk assessment, risk ranking	Technical risk, cost risk, schedule risk, organizational risk, application risk	Requirement analysis and planning phase
16	RIMAM [28]	Risk identification, management, avoidance	Schedule risk and cost risk	
17	SRAEM [22]	Risk estimation	Technical risk, organization risk, environmental risk	
18	SRAEP [24]	Risk assessment, prioritization		
19	SERUM [17]	Implicit and explicit risk management	Generic risk, risk related to planning, development risk	
20	SPRMQ [7]	Risk factor identification, risk probability computation, effects on product quality, risk mitigation and monitoring	Product risks	
21	SITRAM [12]	Risk assessment	Technical risks	
22	Danny [26]	Risk mitigation	Operational risk	Application phase

5 Conclusions

The major contribution of this paper is the SDLC phasewise classification of risks that we have summarized in Sect. 3 using five tables and the consequent mapping of various risk models with different steps of SDLC described in Sect. 4. This systematic study followed by the proposed classification will open up a big horizon for entire risk management process. Researchers will now be able to apply various such models and analyze the occurrence and impact of risks at all steps of SDLC and can mitigate the risk once found meaningful. Generic tools might need to be developed for such purposes with option for tuning them for some specific business model.

References

1. Stern, R., Arias, J.C.: Review of risk management methods. Bus. Intell. J. **4**(1), 59–78 (2011)
2. Silva, P.S., Trigo, A., Varajão, J.: Collaborative risk management in software projects. In: Proceedings of the 8th International Conference on the Quality of Information and Communications Technology, pp. 157–160 (2012)
3. Guiling, L., Xiaojuan, Z.: Research on the risk management of IT project. In: Proceedings of International conference on E-Business and E -Government (ICEE), pp. 1–4, 6–8 May 2011

4. Tianyin, P.: Development of software project risk management model review. In: Proceedings of International Conference on AI, Management Science and Electronic Commerce, pp. 2979–2982 (2011)
5. Avdoshin, S.M., Pesotskaya, E.Y.: Software risk management. In: Proceedings of 7th Central and Eastern European Software Engineering Conference, Russia, pp. 1–6 (2011)
6. Westfall, L.: Software risk management. In: International Conference on Software Quality, San Diego, California, 8–10, February, 2011
7. Mofleh, H.M., Zahary, A.: A framework for software product risk management based on quality attributes and operational life cycle (SPRMQ). http://www.nauss.edu.sa/En/DigitalLibrary/Researches/Documents/2011/articles_2011_3102.pdf
8. Sarigiannidis, L., Chatzoglou, P.D.: Software development project risk management: a new conceptual framework. J. Softw. Eng. Appl. (JSEA) **4**, 293–305 (2011)
9. Sathish Kumar, N., Vinay Sagar, A., Sudheer, Y.: Software risk management—an integrated approach. Global J. Comput. Sci. Technol. (GJCST) **10**(15), 53–57 (2010)
10. Bohem, B.W.: Software risk management: principles and practices. IEEE Softw. **8**, 32–41 (1991)
11. Carr, M.: Taxonomy-based risk identification. Software Engineering Institute, CMU/SEI-93-TR-6 (1993)
12. Loutchkina, I., Jain, L.C., Nguyen, T., Nesterov, S.: Systems' integration technical risks' assessment model (SITRAM). IEEE Trans. Syst. Man Cybern. Syst. **44**(3), 342–352 (2014)
13. Higuera, R.P., Gluch, D.P., Murphy, R.L.: An introduction to team risk management. Special report CMU/SEI, SEI/CMU, Pittsburg, May 1994
14. Alge, B.J., Witheoff, C., Klein, H.J.: When does the medium matter? Knowledge building experiences and opportunities in decision making teams. Organ **91**(1), 26–27 (2003)
15. Mead, N.R.: Measuring the software security requirements engineering process. In: Proceedings of 36th International Conference on Computer Software and Application Workshops, pp. 583–588 (2012)
16. Kontio, J., Basili, V.R.: Empirical evaluation of a risk management method. In: SEI Conference on Risk Management, Atlantic City (1997)
17. Greer, D.: SERUM: software engineering risk: understanding and management. J. Proj. Bus. Risk Manag. **1**(4), 373–388 (1997)
18. Hillson, D.A.: Towards risk maturity model. Int. J. Proj. Bus. Risk Manag. **1**(1), 35–45 (1997). ISSN:1366-2163 (Spring)
19. Foo, S.-W., Muruganatham, A., Software risk assessment model. ICMIT 2000, IEEE, pp. 536–544
20. Keshlaf, A.A., Hashim, K.: A model and prototype tool to manage software risks. In: Proceedings of the 1st Asia-Pacific Conference on Quality Software (APAQS'00), Washington, DC, USA (2000)
21. Armestrong, R., Adens, G.: Managing Software Project Risks. TASSC Technical paper, USA (2008). www.tassc-solutions.com. January 2008 Copyright 2001–2010, Tassc Limited
22. Gupta, D., Sadiq, M.: Software risk assessment and estimation model. In: International Conference on Computer Science and International Technology, pp. 963–967. IEEE Computer Society, Singapore (2008)
23. Rabbi, M., Mannan, K.: A review of software risk management for selection of best tools and techniques. In: Proceedings of 9th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, pp. 773–778 (2008)
24. Sadiq, M., Rahmani, M.K.I., Ahmad, M.W., Jung, S.: Software risk assessment and evaluation process (SRAEP) using model based approach. In: International Conference on Networking and Information Technology (ICNIT), pp. 171−177 (2010)
25. Hoodat, H., Rashidi, H.: Classification and analysis of risks in software engineering. In: WASET-2009, pp. 446–452
26. Danny, L.: Reducing operational risk by improving production software quality. Softw. Risk Reduction Rev. **13**, 1–15 (2013)

27. Suebkuna, B., Ramingwong, S.: Towards a complete project oriented risk management model: a refinement of PRORISK. In: Eighth International Joint Conference on Computer Science and Software Engineering (JCSSE), pp. 349–354, 11–13 May 2011
28. Shahzad, B., Al-Ohali, Y., Abdullah, A.: Trivial model for mitigation of risks in software development life cycle. Int. J. Phys. Sci. **6**(8), 2072–2082 (2011)
29. Roy, G.G.: A risk management framework for software engineering practice. In: Proceedings of the Australian Software Engineering Conference (AAWEC'04) (2014)
30. Amber, S., Shawoo, N., Begum, S.: Determination of risk during requirement engineering process. Int. J. Emerg. Trends Comput. Inform. Sci. **3**(3), 358–364 (2012)
31. Pandey, D., Suman, U., Ramani, A.K.: Security requirement engineering issues in risk management. Int. J. Comput. Appl. **17**(5), 11–14 (2011)
32. Islam, S., Houmb, S.H.: Integrating risk management activities into requirements engineering. RCIS-2010, Nice, France, May 2010, pp. 299–310
33. Drs. Erik, P.W.M.: Practical risk-based testing—product risk management: the PRISMA method, EuroSTAR-2011, Manchester, UK, pp. 1–24, 21–24 November 2011
34. Kwan, T.W., Leung, H.K.N.: A risk management methodology for project risk dependencies. IEEE Trans. Softw. Eng. **37**(5), 635–648 (2011)
35. Lobato, L.L., Neto, S., da Mota, P.A., do Carmo Machado, I.: A study on risk management for software engineering. In: Proceedings of the EASE, pp. 47–51 (2012)
36. Lobato, L.L., da Mota, P.A., Neto, S., do Carmo Machado, I., de Almeida, E.S., de Lemos Meira, S.R.: Evidence from risk management in software product lines development: a cross-case analysis. In: Proceedings of 6th Brazilian Symposium on Software Components, Architectures and Reuse (2012)
37. Nolan, A.J., Abrahão, S., Clements, P.C., Pickard, A.: Requirements uncertainty in a software product line. In: Proceedings of 15th International Software Product Line Conference, pp. 223–231 (2011)
38. Lobato, L.L. et al.: Risk management in software product lines: an industrial case study. In: Proceedings of ICSSP, Switzerland, pp. 180–189 (2012)
39. Gonzalo, E., Gallardo, E.: Using configuration management and product line software paradigms to support the experimentation process in software engineering. RCIS-2012, Valencia, May 2012. pp. 1–6
40. Khoo, Y.B., Zhou, M., Kayis, B., Savci, S., Ahmed, A., Kusumo, R.: An agent-based risk management tool for concurrent engineering projects. Complex. Int. **12**, 1–11 (2008)
41. Sharif, A.M., Rozan, M.Z.A.: Design and implementation of project time management risk assessment tool for SME projects using oracle application express. In: World Academy of Science, Engineering, and Technology (WASET), vol. 65, pp. 1221–1226 (2010)
42. Snekir, W.G., Walker, P.L.: Enterprise risk management: tools and techniques for effective implementation. Institute of management Accounts, pp. 1–31 (2007)