STUDY OF EXISTING RISK MANAGEMENT MODELS AND PRIOR RESEARCH CONTRIBUTION

***Dr. L. Manjunath Rao**
Prof.: Dept. of Computer Science
Dr. Ambedkar Institute of Technology, Bangalore, India

****Salma Firdose**
Research Scholar, Research and Development Centre,
Bharathiar University, Coimbatore, India

ABSTRACT

Different software projects development methodologies exist in current era, however, selecting the methodology that most closely fits a computer code depends on many factors. One necessary issue is the extents of how much risky the project is. Another issue is that the degree to that every methodology supports risk management. Indeed, the literature is wealthy in such studies that aim at scrutiny the presently out there computer code development method models from totally different views. In distinction, very little effort has been spent in purpose of scrutiny the out there method models in terms of its support to risk management. During the discussion, we tend to investigate the state of risk and risk management within the most well-liked computer code development method models (i.e. waterfall, v-model, progressive development, spiral, and agile development). This trend in such studies is anticipated to serve in many aspects. Technically, it helps project managers adopt the methodology that most accurately fits their projects. From another facet, it'll build the simplest way for additional studies that aim at up the computer code development method.

Keywords *- component; Software Engineering, Risk Management*

I. INTRODUCTION

Risk management in software engineering is related to the various future harms that could be possible on the software due to some minor or non-noticeable mistakes in software development project or process. “Software projects have a high probability of failure so effective software development means dealing with risks adequately (www.thedacs.com).” Risk management is the most important issue involved in the software project development. This issue is generally managed by Software Project Management (SPM). During the life cycle of software projects, various risks are associated with them. These risks in the software project is identified and managed by software risk management which is a part of SPM.. The recent study unconcealed that solely tierce of software package developments are often thought-about thriving [1]. This means that software package projects’ failure rate remains intolerably high, that might be attributed to the magnified quality of software package development projects besides the absence or the poorly-applied risk management method. so as to attain project success, the researchers believe that the most effective contribution to manage risks in software

package projects is to pick the foremost appropriate methodology that most closely fits the meant project, and to think about it throughout the event method as a mean to manage risks. A software package development methodology or a software package development method model is an approach to the software Development Life Cycle (SDLC) that describes the sequence of steps to be followed whereas developing software package projects [2]. Several software package development methodologies exist, they disagree from one another in terms of your time to unharnessed, quality, and risk management. in spite of the followed methodology, the fundamental lifecycle activities are enclosed altogether lifecycle models, however in all probability in several orders. These models could be consecutive (i.e. waterfall) or unvaried (i.e. evolutionary). they could be specification-driven (i.e. waterfall), code-driven (i.e. evolutionary), or risk-driven (i.e. spiral). Moreover, they could be typical (i.e. conventional waterfall) or agile (i.e. scrum). Indeed, there's no ideal model that matches all the software package development projects; for sure circumstances every model has its benefits and drawbacks. Deciding upon the methodology to follow depends on the event setting, the kind of the project underdevelopment, the event team, and also the potential risks. Thus, it falls on behalf of the developer to pick the methodology (or any tailored combination) that most closely fits the project circumstances [3]. Because the potential risks in any software package project greatly influence the choice of the foremost applicable software package development methodology, risk management is presently thought-about the foremost goal of any elite methodologies. Hence, any software package development methodology is best enforced if it's thought-about as a mean to manage risks. Totally different software package development methodologies support risk management naturally in variant levels. Within the following sections we have a tendency to investigate the state of risk management within the most typical software package development methodologies. The analysis methodology followed was a scientific literature review that failed to chiefly aim at examination the prevailing software package development methodologies, rather to conduct this comparative study between these models with reference to their illustration of risk management. The prime objective of this investigation is to discuss a body of proof that's risk management is a common part of software package development methodologies along with the risk-driven ones [4].

II. RISK MODEL STUDY

In this section we review the leading software development methodologies (i.e. waterfall, V-model, incremental, spiral, and agile) and investigate the state of risk management in each of these models. For each one, we highlight the sources of risks it came to resolve, and uncover the risky areas hindering its implementation.

Waterfall Model:- It was first introduced but not named by Royce in 1970. It abstracts the essential software development activities (i.e. requirements, analysis, design, coding, testing, and operation) in a sequential manner. Waterfall development was proposed to avoid the risks introduced by the code and fix technique by

inserting the requirements and analysis stages before the coding stage. This ensures that user requirements are clearly defined in advance, thus, reduces the time and effort wasted on several iterations of code and fix. In the original waterfall model, any error occurs at any stage propagates into the subsequent stages until it is lately discovered in the testing phase. To avoid this risk, Royce [5] suggested that at the beginning of each stage a review to the previous stage should be conducted to ensure that the previous stage was properly done. Later, he modified his original waterfall model by adding localized iterations that provide feedback to the previous phases. However, even with these localized iterations, problems are still being discovered in the testing phase, these problems are usually due to problems in the design stage or in the requirements stage. Thus, to recover from these errors, complex iterations to the design stage and to the requirements stage were added. These iterations consume a lot of time, efforts, and other resources. In order to avoid the risks of the operational constraints, Royce [5] suggested a preliminary design phase to be inserted between the requirements phase and analysis phase in order to impose constraints on the analysts. This is properly accomplished by the iterative loop between the preliminary design and the analysis stages until a satisfactory preliminary design is reached.

- **Major Sources of Risk in the Waterfall Model:-** From the above discussion, we can conclude that risks in the waterfall model are unavoidable, even in the Royce's modified waterfall model; this is due to the nature of the model itself. The major sources of risk in the waterfall model are listed below:

- **Continuous requirements change:-** The major risk factor threatens the waterfall projects is the continuous requirements change during the development process. The waterfall model cannot accommodate with these changes due to its strict structure. The waterfall model requires that all requirements be clearly defined in advance in the requirements stage in order to guarantee that no change could appear later on during the development process. Clearly, this is an idealistic situation, since it is difficult for the real projects to identify all requirements previously. Thus, it is even impossible to guard requirements from being changed. Actually, continuous requirements change is not a problem to be solved, neither it is restricted exclusively to the waterfall model. Rather, it is the unstable nature of the software projects besides the highly strict nature of the waterfall model what made its consequences significant in the waterfall model mainly.

- **No overlapping between stages:-** Another source of risk in the waterfall model is that it requires each stage to be completed entirely before proceeding into the subsequent phase. In other words, it does not allow overlapping between stages. Obviously, this will waste time, cost and other resources, since the stages in the waterfall model are relatively long. Hence, most team members who are responsible for specific stages will spend most of their time waiting for other stages to complete so that they can start doing their work.

- **Poor quality assurance:-** Lack of quality assurance during the different phases of the development process is another source of risk. Validating the product is restricted to a single testing phase lately in the development process. Hence, the testing phase in the waterfall model is the highest risky phase, since it is the last stage wherein the system is put as a subject for testing. Thus, all problems, bugs, and risks are discovered too late when the recovering from these problems requires large rework which consumes time, cost, and effort.
- **Relatively long stages:-** Another source of risk in this model resides in the relatively long stages, which makes it difficult to estimate, time, cost, and other resources required to complete each stage successfully. Additionally, in the waterfall model, there is no working product until late in the development process when the product is almost complete and any change is impossible. To make things worse; imagine if the product failed to meet users' expectations.

Incremental development: Incremental development is a variant of the waterfall model which consists of a series of waterfall lifecycles wherein the software development project is broken down into smaller segments called increments. The proposal of the incremental development was to accommodate with risks inherent from implementing the overall software project over a single lifecycle in the pure waterfall model. First of all, since the project is broken down into smaller segments, the development effort is distributed among several increments. Thus, risks are spread over multiple iterations rather than single iteration as in the pure waterfall development. Clearly, it would be easier to manage those risks in the former case. The major risk factor threatens the waterfall development is that it requires all requirements be clearly defined in advance, since its structure does not allow requirements to be changed during the development process. The incremental development reduces this risk by grouping requirements, then implementing each group in an increment repeatedly until the system is complete and all requirements are met. Despite the fact that most requirements have to be known in advance, building requirements incrementally allows new requirements to be added later on in subsequent increments. The incremental development also allows requirements to be changed; these changes are reflected in the subsequent increments. Changing requirements comes after a feedback from the customer about the already developed increments which can be considered as prototypes for the subsequent increments. The other risk of the waterfall reduced by the incremental development is the time, cost, and other resources wasted from prohibiting overlapping. The incremental development allows many mini increments to overlap, thus most team members can work in parallel. Errors in the previous increments could be fixed during the development of the current increment. Obviously, this saves time, cost, and other resources. Thus, the initial deadlines are more likely to be met. Unlike the waterfall model, the incremental development allows initial releases with core functionality to be delivered to the customer early. Indeed, these releases are working non-completed systems delivered early

to the customers in order to help them build a realistic impression about the system underdevelopment, and to enable them to give their feedback early so that the cost of any change would be as less as possible. Another issue related to the user acceptance of the system; the system would be more acceptable if it is introduced to the end users gradually bit by bit instead of introducing differently new system at once as in the waterfall model [6]. Still, the incremental development suffers from different sources of risks that are illustrated below:

- Delayed requirements implementation:- One major risk of the incremental model resides in that developers tend to postpone requirements, so that they are included later on in subsequent increments. Obviously, this risk factor should be avoided, since the delayed requirements might be core ones upon which the user acceptance of the whole system depends. Thus, it is recommended that all identified requirements be addressed in the initial increments of the system, and the later increments should be left for any newly identified requirements or any change in the previously defined ones.
- Propagation of bugs through increments:- Another source of risk is that letting any undiscovered bug in one increment to propagate through subsequent increments. It is easier to repair from bugs in the earlier increments of the development, while it might be much more difficult or even impossible after the system enlarges. This might be due to poor testing and maintenance process conducted at the end of each increment.
- **Underestimation of time and other resources required for each increment:-** The inadequate estimation of time, cost, and other resources required for each increment also affects the project underdevelopment. The underestimation of time required for each increment delays the implementation of the subsequent increments. This delay results in an unmet project deadlines. This inadequate estimation might cause time contention wherein either extra burden is put on the shoulders of developers, or some requirements be ignored.
- **Time and cost overrun:-** Time and cost overrun is a critical factor too. This deadly interrupts the development process. Despite the fact that any interrupt at any point in the incremental development process results in a working system, mostly this system would be an uncompleted system wherein some functionalities are not implemented yet.

V-Model: As discussed before, one of the major risk factors threaten the waterfall model is the poor verification and validation methods, which are restricted to a single testing phase conducted lately in the development process. Another variant of the waterfall model that came out to deal with this risk is the V-model. The V-model is a testing-focused software development process. It gives equal importance to both development and testing. Its symmetrical shape allows the testing process to start early at the development process, and to be aligned with its different phases. This could be achieved by designing test plans and test cases during each development phase prior to the actual testing;

this allows requirements and designs to be verified easily during the corresponding testing phases. Moreover, test planning conducted at each stage helps at early identification of project's specific risks and reducing them through an improved process management. Another enhanced version of the V-model is the V+ model; it adds user involvement, risk, and opportunities to the z-axis of the V-model. Although the V-model is a highly structured, well disciplined process model, today's developers think of it as a too rigid process model due to the inflexibility it exhibits against the current evolutionary nature of software projects [7].

Spiral Development: The spiral model was proposed by Boehm [8] in 1988 as a risk-driven software development process model, wherein the whole development process is guided by the involved risks. It aims at identifying and evaluating software project risks, and helps in reducing these risks and controlling project cost in a favour of a better controlled software project. Indeed, the explicit risk management in spiral distinguishes it among other process models which employ some kinds of risk management as subtasks; without this level of the explicit representation as in spiral [9]. In spiral, this feature guarantees that most risks are recognized early and much earlier than it is in other process models. Spiral development supports risk management in software projects in several ways summarized in the following:

- The initial risk analysis that acts as a look-ahead step and aims at:
- Identifying most risks threaten the project.
- Classifying risks into user interface risks and development risks
- Evaluate these risks to decide upon the risks to handle through each cycle. Moreover this classification helps developers in implementing risk resolution techniques such as prototyping and benchmarking.

The evolutionary prototyping spirals that aim at resolving performance and user interface related risks. These spirals help in reducing major risks before proceeding into the development process.

- The risk analysis stage at each cycle that precedes each phase of the waterfall phases in purpose of:
- Resolving program development and interface control risks inherent from the start of the project.
- Evaluating and resolving the new risks that might arise after changing any of the objectives, alternatives, or constraints at the beginning of the cycle.
- The iterative feature of the spiral which allows the development process to go back to the first quadrant at any point in progress which allows:
- Objectives, alternatives and constraints to change as more attractive alternatives exist.
- New technology to be incorporated easily during the development process.

The review conducted at the end of each cycle with main stakeholders as a decision point to avoid the lack of commitment risks during the next cycle. Time and cost overrun risks are best managed using spiral development due to the

risk analysis stage conducted at each cycle. In this stage, the cost and time required for each cycle are analyzed in advance to give a clear picture about the critical state of the project. This helps the project manager and the developers get more control over these risks. Risks related to the increased complexity of the project are also managed using spiral. This is achieved by the partitioning activity conducted at the planning phase. Decomposing the project into portions to be developed in parallel spirals obviously reduces time contention related risks, since more work could be achieved during the same interval. Despite its risk driven nature, spiral has its own sources of risks which are summarized in the following:

- **High reliance on the human factor:-** All the activities related to identifying, analyzing, and resolving risks rely on the experience of developers and their abilities in identifying and managing risks [7]. If these abilities are unavailable, major risks might remain hidden for several lifecycles and discovered late when it matured into real problems. At that time, the cost of rework to recover from these risks becomes very high.
- **Detailed Risk Management Process:-** Cost and schedule risks might increase using spiral due to its iterative feature, especially for low risk projects wherein risk assessment is not required to be at this level of granularity.

Agile Development: Agile is a term first introduced in 2001 to refer to a group of lightweight software development methodologies evolved in the mid-1990s including Scrum (1995), Crystal Clear, Extreme Programming (1996), Adaptive Software Development, Feature Driven Development, and Dynamic Systems Development Method (DSDM) (1995) [10]. In contrast to the heavyweight methodologies (i.e. waterfall), the lightweight methodologies deemphasize a formal process step; they proceed in the development process without waiting for formal requirements and design specifications. The main point that the agile focuses on is the close, Informal communication between the different system stakeholders including the developers and the customer representative. Indeed, in agile, this communication is the source of planning, requirements, identifying risks, feedback, and changes. Building upon the literature, we can say that there are two contrasting views regarding risk management in the agile context. The first claims that agile is an inherent risk driven approach and implicitly supports risk management by nature. The proponents believe that there is no need to enhance risk management in these projects. In contrast, the second [11] believes that the risk management state in agile does not differ significantly from other traditional models and that risk management should be enhanced in agile to compensate for the lack of risk management in the agile projects. The advocates to the second view believe in that in some situations the inherent risk management driven nature of the agile is insufficient [12]. As mentioned before, the major risk factor threatens today's software projects is the continuous changes it faces in requirements and the surrounding environment. The agile development addresses this risk. The agile is an adaptive approach; it exhibits a flexible response

to change, this is due to the incremental, iterative approach it adapts, wherein each increment is very short and the developers are in a continuous interaction with the customer. Thus, any change in requirements will be discovered early as soon as the software first releases are produced, then the project can adapt to these changes quickly. Due to the close frequent interaction with the customer, requirements are collected during each increment directly from the customer rather than from formal documents that represent them as in other traditional development methods. This would eliminate any ambiguity in understanding requirements, and ensure stakeholders' commitments to the requirements they provide. Agile development best fits software projects which lack structured planning, due to its adaptive planning feature which requires minimal planning activities be conducted formally. Using agile development, the risk of delivering software that contains bugs will be reduced due to its reliance on automated test cases [13]. Thus, the software is tested at each release, and retested again if a bug was discovered to make sure that it has been eliminated. In spite of the assertions it makes regarding managing risks, the agile development lacks for any detailed suggestions for managing these risks. Thus, many sources of risks will be left unhandled. The following are the major sources of risk in the agile development:

- **Very large software system:-** The inherent risk management in agile development is not sufficient for large, complex software systems, since the resulting increments would be relatively large. This would increase the time span between increments, and thus require a higher cost to deal with changes and bugs if discovered.
- **Large development team:-** It is not suitable for large teams, since managing the communication between their members would be much more difficult.
- **High reliance on human factor:-** It relies entirely on the experience of the development team and their abilities to communicate successfully with customers. If the project misses these conditions, then the failure is an inevitable issue.
- **Inappropriate customer representative:-** The unavailability of an appropriate customer representative is another risk factor. Actually, this factor influences the development process as much as team members' factor.
- **Distributed development environment:-** This approach is not suitable for developing software projects in distributed environment, since it requires a close face to face interaction communication between the development team. Else, other communication methods such as video conferencing should be held at daily basis.
- **Scope creep:-** Another important risk factor is the scope creep, this usually happens due to the minimal planning conducted in this methodology which causes developers to become distracted from the project main objectives. As a result, the project will enlarge, become more complex, and finally the project will overrun.

III. PRIOR RESEARCH WORK

Mofleh and Zahary [14] presented a framework that tries to improve software product risk management by applying some sequential processes during operational life cycle of the product. The framework is called SPRMQ (a framework for Software Product Risk Management based on Quality attributes and operational life cycle) which attempts to manage software product risk.

Sarigiannidis et al. [15] investigated a wide range of relevant literature, proposes a new conceptual framework for managing risk in software development projects, intro-duces new conceptual factors, brings out their interrelation, and suggests new prospects and managerial implications for both practitioners and academics

Kipyegen et al. [16] developed a framework that guides in the adoption of the existing formal risk management techniques in two areas; Institutions of learning and software development industry.

Elzamly et al. [17] proposed the new framework software risk management methodology for successful software project. There are five main phases such as identification risk, risk analysis and evaluation, risk treatment, risk controlling, risk communication and documentation for software development life cycle. Indeed, our approach focuses on identifying software risk factors, and risk management techniques and on how to manage software risk factors with statistical and mining techniques.

Conforti et al. [18] presented an innovative framework for process-related risk management and describes a working implementation realized by extending the YAWL system. The framework covers three aspects of risk management: risk monitoring, risk prevention, and risk mitigation.

Bannerman et al. [19] introduced variations in the risk and project management challenges they face. Findings also suggest that formal project management is neither necessary nor sufficient for project success.

Roy [20] provided a brief introduction to the concepts of risk management for software development projects, and then an overview of a new risk management framework.

Keshlaf et al. [21] demonstrated number of software risk management approaches and identify weaknesses such as the treatment of culture issues, geographical location, and process and product perspectives.

IV. CONCLUSION

In this paper we have reviewed the leading software development process models and investigated the state of risk management in each of these models. As a result, we found that some software development methodologies inherently involve risk management. For each methodology, this requires certain circumstances to exist. This indicates that risks are inevitable in most software development methodologies, and that all software development methodologies, including the risk-driven ones, require that risk management be enhanced in it. An interesting dimension for future research is to find out a strategy that aims at enhancing risk management in the different software development methodologies.

REFERENCES

[1] The Standish Group, “Report CHAOS”, Project Smart, 2014

[2] “The Software Development Life Cycle (SDLC)”,Pilican Engineering, Dcoument ID-0-2, Version 2.0

[3] Jones and Bartlett “Software Process Models”,LLC, 2004

[4] H.Hijazi, T.Khdour, A.Alarabeyyat, “ A Review of Risk Management in Different Software Development Methodologies”, International Journal of Computer Applications (0975 – 8887) Volume 45– No.7, May 2012

[5] W. Royce, “Managing the development of large software systems,” IEEE WESCON, pp. 1-9,, 1970

[6] R. Flask “The System Life Cycle”, Charlie Abela

[7] “Software Development Life Cycle (SDLC)”, Tuorial Points

[8] B.W. Boehm, "A spiral model of software development and enhancement." Computer 21, no. 5, pp. 61-72, 1988

[9] “NASA Risk Management Handbook” NASA/SP-2011-3422 Version 1.0, 2011

[10] “An Introduction to Agile Software Development”,Serena, 2007

[11] Jakub Miler “A Method of Software Project Risk Identification and Analysis”, Gdańsk University of Technology, 2005

[12] Schmietendorf, Andreas, André Scholz, and Claus Rautenstrauch. "Evaluating the performance engineering process." Proceedings of the 2nd international workshop on Software and performance, ACM, 2000

[13] Wallmüller, E. "Risk management for IT and software projects." In Business continuity, pp. 165-178. Springer Berlin Heidelberg, 2002

[14] M.HALIMA and A.Zahary. "A Framework For Software Product Risk Management Based On Quality Attributes And Operational Life Cycle (SPRMQ)." Risk 1(3), 2010

[15] L. Sarigiannidis and P. D. Chatzoglou. "Software development project risk management: A new conceptual framework." Journal of Software Engineering and Applications, Vol.4, No. 05, 293, 2011

[16] N.J.Kipyegen, W.Mwangi, and S. t.Kimani. "Risk Management Adoption Framework for Software Projects: A Case Study for Kenyan Software Project Managers and Developers." International Journal of Computer Science Issues (IJCSI), Vol.9, No. 3, 2012

[17] A. Elzamly and B.Hussin. "AN Enhancement of Framework Software Risk Management Methodology for Successful Software Development" Journal of Theoretical & Applied Information Technology, Vol. 62, No. 2,2014

[18] R. Conforti, M.L.Rosa, A.H.M. T.Hofstede, G.Fortino, M.D.Leoni and M.J. Adams. "A software framework for

risk-aware business process management." In Proceedings of the CAiSE'13 Forum at the 25th International Conference on Advanced Information Systems Engineering (CAiSE): CEUR Workshop Proceedings, Vol. 998, pp. 130-137. Sun SITE Central Europe, 2013.

[19] P.L. Bannerman "Risk and risk management in software projects: A reassessment." Journal of Systems and Software, Vol. 81, No. 12, pp.2118-2133, 2008

[20] G.G.Roy"A risk management framework for software engineering practice." In Software Engineering Conference, 2004. Proceedings. 2004 Australian, pp. 60-67, 2004

[21] A. K. Ali, A. Ali, and S.Riddle. "Risk management for web and distributed software development projects." In Internet Monitoring and Protection (ICIMP), 2010 Fifth International Conference, pp. 22-28, 2010

[22] Efficient Framework of e-Government for Mining Knowledge from Massive Grievance Redressal Data, G Sangeetha, LM Rao- International Journal of Advanced Research in Communication Engineering,2015

[23] 'Branding Strategy-Delivering the brand promise in a competitive environment' GJ-MIT, Mohali, India