A Review of Risk Management in Different Software Development Methodologies

Haneen Hijazi
Hashemite University
Zarqa, Jordan

Thair Khdour
Al Balqa Applied University
Salt, Jordan

Abdulsalam Alarabeyyat
Al Balqa Applied University
Salt, Jordan

ABSTRACT
Different software development methodologies exist. Choosing the methodology that best fits a software project depends on several factors. One important factor is how risky the project is. Another factor is the degree to which each methodology supports risk management. Indeed, the literature is rich in such studies that aim at comparing the currently available software development process models from different perspectives. In contrast, little effort has been spent in purpose of comparing the available process models in terms of its support to risk management. In this paper, we investigate the state of risk and risk management in the most popular software development process models (i.e. waterfall, v-model, incremental development, spiral, and agile development). This trend in such studies is expected to serve in several aspects. Technically, it helps project managers adopt the methodology that best suits their projects. From another side, it will make a way for further studies that aim at improving the software development process.

General Terms
Software Engineering, Risk Management.

Keywords
Risk; Risk management; Software development process model; Software development methodology; Waterfall; Spiral; Incremental; V-model; Agile.

1. INTRODUCTION
The recent report by Standish group in 2009 revealed that only one-third of software projects can be considered successful [1]. This implies that software projects' failure rate remains unacceptably high, which could be attributed to the increased complexity of software development projects besides the absence or the poorly-applied risk management process.

In order to achieve project success, we believe that the best way to manage risks in software projects is to select the most suitable methodology that best fits the intended project, and to consider it during the development process as a mean to manage risks.

A software development methodology or a software development process model is an approach to the Software Development Life Cycle (SDLC) that describes the sequence of steps to be followed while developing software projects [2, 3].

Many software development methodologies exist, they differ from each other in terms of time to release, quality, and risk management. Regardless of the followed methodology, the basic lifecycle activities are included in all lifecycle models, but probably in different orders. These models might be sequential (i.e. waterfall) or iterative (i.e. evolutionary). They might be specification-driven (i.e. waterfall), code-driven (i.e. evolutionary), or risk-driven (i.e. spiral). Moreover, they might be conventional (i.e. traditional waterfall) or agile (i.e. scrum).

Indeed, there is no ideal model that fits all the software development projects; in certain circumstances each model has its advantages and disadvantages. Deciding upon the methodology to follow depends on the development environment, the type of the project underdevelopment, the development team, and the potential risks. Thus, it falls on behalf of the developer to select the methodology (or any customized combination) that best fits the project circumstances [4].

As the potential risks in any software project greatly influence the selection of the most appropriate software development methodology, risk management is currently considered the major goal of any selected methodologies. Hence, any software development methodology is best implemented if it is considered as a mean to manage risks.

Different software development methodologies support risk management by nature in variant levels. In the following sections we investigate the state of risk management in the most common software development methodologies. The research method followed was a systematic literature review that did not mainly aim at comparing the existing software development methodologies, rather to conduct this comparative study between these models with respect to their representation of risk management. The main objective of this investigation is to present a body of evidence that is risk management is need in all software development methodologies even the risk-driven ones.

2. RELATED WORK
Much work has been done around the field of software development methodologies. Most of these studies are comparative analysis between these methodologies from different perspectives. Indeed, the literature lacks such studies that conduct the comparative analysis in terms of risk management. In 1996, Sommerville [4] reviewed the dominant models and concluded that the existence of an ideal model that suits all projects is unrealistic. Guimaraes and Vilela [2] in 2005 compared between the waterfall and the spiral using a more systematic approach called "Compare Development Model" (CDM).The comparable components used in their study were objectives, requirements, analysis, procedures, project, tests, and the operation. In 2008, Shahzad et al. [5] discussed the major factors that the project being developed may encounter using the incremental model. Rodrguez et al. conducted a descriptive comparative study in 2009 between the SDLC process models; they used a Meta model and suggested that each model should be an instance of it. In 2008, Nyfjord and Kajko- Mattsson [7] conducted a comparative study between the waterfall and the iterative incremental development (IID), they adopted three parameters

in their comparison that are cost, duration, and comparison. In 2010, Dash and Dash [8] discussed the waterfall model and its exposure to risks throughout the SDLC. In the same year, Ruparelia [3] reviewed the most popular software development process models in terms of the application types each fits. Also in 2010, Munassar and Govardhan [9] conducted a comparative study between the dominant methodologies, illustrated their phases, advantages and disadvantages, and how they differ from each other.

3. ANALYSIS

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.

3.1 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 [10] 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 [10] 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!

3.2 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 [11].

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 [12].

Major Sources of Risk in the Incremental Development

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.

3.3 The 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 [3].

3.4 Spiral Development

The spiral model was proposed by Boehm [13] 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 [14]. 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 maximum optimization of project resources usage.
  - To deal with poorly done activities in the earlier phases.
- 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.

Major Sources of Risk in the Spiral Model

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 [13]. 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.

3.5 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) [15]. 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 [16] 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 [17].

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 [18]. 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 [19]. Thus, the software is tested at each release, and retested again if a bug was discovered to make sure that it has been eliminated.

Major Sources of Risk in the Agile Development

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.

4. CONCLUSION AND FUTURE WORK

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.

5. REFERENCES

[1] Standish Group, "CHAOS report," 2009, Boston.

[2] L. Guimares and P. Vilela, "Comparing Software Development Models Using CDM," Proceedings of The 6th Conference on Information Technology Education, New Jersey, 20-22 October 2005, pp. 339-347.

[3] N. Ruparelia, "Software Development Lifecycle Models," ACM SIGSOFT Software Engineering Notes, Vol. 35, No. 3, 2010, pp. 8-13.

[4] I. Sommerville, "Software process models," ACM Computing Surveys, Vol. 28, No. 1, 1996, pp. 269-271.

[5] B. Shahzad and S. Safvi, "Effective Risk Mitigation: A User Prospective," International Journal of Mathematics and Computers in Simulation, Vol. 2, No. 1, 2008, pp. 70-80.

[6] L. Rodrguez, M. Mora, and F. Alvarez, "A descriptive Comparative Study of the Evolution of Process Models of Software Development Lifecycles (PM-SDLCs)," Proceedings of the Mexican International Conference on Computer Science, 2009, pp. 298–303.

[7] J. Nyfjord and M. Kajko-Mattsson, "Outlining A Model Integrating Risk Management and Agile Software Development," Proceedings of the 34th Euromicro Conference Software Engineering and Advanced Applications, 2008, pp. 476-483.

[8] R. Dash and R. Dash, "Risk assessment techniques for software development," European Journal of Scientific Research, Vol. 42, No. 4, 2010, pp. 629–636.

[9] N. Munassar and A. Govardhan, "A Comparison between Five Models of Software Engineering," International Journal of Computer Science Issues (IJCSI), Vol. 7, No. 5, 2010, pp. 94–101.

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

[11] GSAM, "Condensed GSAM Handbook, chapter 2: Software Life Cycle," 2003.

[12] G. Tate and J. Verner, "Case Study of Risk Management, Incremental Development and Evolutionary Prototyping," Information and Software Technology, Vol. 32, No. 3, 1990, pp. 207-214.

[13] B. Boehm, "A Spiral Model of Software Development and Enhancement," Computer, 1988, pp. 61-72.

[14] B. Gotterbarn, "Enhancing risk analysis using software development impact statements," Proceedings of the 26th Annual NASA Goddard Software Engineering Workshop, 2001, pp. 43-51.

[15] V. Szalvay, "An Introduction to Agile Software Development," technical report, Danube Technology, 2004.

[16] J. Miller and J. Grski, "A Method of Software Project Risk Identification and Analysis," Ph.D. Thesis, Faculty of Electronics, Telecommunications and Informatics, Gdansk University Of Technology, 2005.

[17] A. Schmietendorf, E. Dimitrov, and R. Dumke, "Process Models for the Software Development and Performance Engineering Tasks," Proceedings of the 3rd International Workshop on Software and Performance, 2002, pp. 211-218.

[18] F. Nasution and R. Weistroffer, "Documentation in Systems Development a Significant Criterion for Project Success," Proceedings of the 42nd Hawaii International Conference on System Sciences, 2000, pp. 1-9.

[19] S. Murthi, "Preventive Risk Management for Software Projects," IT Professional, Vol. 4, No. 5, 2002, pp. 9-15.