Asian Journal of Research in Computer Science

14(3): 12-24, 2022; Article no.AJRCOS.84911
ISSN: 2581-8260

Explicit Risk Management in Agile Software Projects: Its Relevance and Benefits

Osaki Miller Thom-Manuel [a*]

[a] *Information and Communication Technology Centre, Ignatius Ajuru University of Education, Nigeria.*

Author's contribution

The sole author designed, analysed, interpreted and prepared the manuscript.

Article Information

DOI: 10.9734/AJRCOS/2022/v14i330340

Open Peer Review History:
This journal follows the Advanced Open Peer Review policy. Identity of the Reviewers, Editor(s) and additional Reviewers, peer review comments, different versions of the manuscript, comments of the editors, etc are available here: https://www.sdiarticle5.com/review-history/84911

Review Article

Received 23 March 2022
Accepted 29 May 2022
Published 26 August 2022

ABSTRACT

Risk management is the systematic process of controlling risks and it is critical to the success of all software development projects. Agile software development methodologies by the inbuilt features utilized, control risks. However, this does not work for all cases of software development projects and supplementary means may need to be applied. This can be addressed by introducing an intentional and formal way of managing risks in the agile environment. Explicit risk management promises numerous benefits if properly implemented.

This study intends to review and deduce based on the risks identified, the relevance and benefits of formally managing risks in agile software development projects. To achieve the aim of this study, the researcher reviewed risk management procedures in a typical agile setting as well as research that exposed the insufficiency of the inherent risk management process in agile projects and the identified risks. Related research papers from peer-reviewed journals and other reliable sources were reviewed to extract risks that occurred using agile without explicit risks management. The study inferred that some risks do exist that occurred with the introduction and use of the agile method itself. Also, there could be risks that surface when the project size exceeds a limit. Thus, managing risks explicitly will go a long way to address such risks. Consequently, the researcher was able to deduce the relevance and benefits of implementing explicit risk management in an agile software development project.

This study showed that it is beneficial to incorporate formal risk management procedures in agile software development when mega software projects are being developed. However, to maintain the agility of the agile methods which happens to be a major benefit of the utilization of agile methods, more research is needed to further explore explicit risk management in the agile environment without violating the swiftness in the agile settings.

**Corresponding author: E-mail: osaki.miller@iaue.edu.ng, osai.miller@iaue.edu.ng;*

Keywords: Explicit risk management; agile; software projects; SCRUM; risk.

1. INTRODUCTION

Like every other project, software development projects are faced with some level of uncertainty. Uncertainties in software projects may result in risks ranging from scope creep, budget overrun [1], not meeting the scheduled delivery deadline, production of software with wrong and or incomplete software specifications [2], and many more.

Risks are inevitable in every project and must be managed. Risk management is the systematic procedure that involves identifying, analyzing, prioritizing, controlling, and monitoring risks to reduce and possibly eliminate any negative outcome of the risks in actualizing the project's objectives. It is a must-follow process [3,4] and needs to be carried out throughout the entire software development process. The need for managing risks even becomes more necessary as the project size and complexity increase [5]. The significance of risk management in software development has led to the proposition of several standards that are generic namely ISO 31000 [6], ISO 14971 [7], and PMBOK [8].

Agile software development methodologies are a group of methods having some common features namely time-boxed development cycles, ability to change or modify requirements during development, constant communication with customers, involvement of self-organizing cross-functional teams, and regular testing after each iteration builds among others. These inbuilt features manage software development risks and especially internal or project-individual risks [9]. To manage external risks, some form of formal procedure is needed to identify, analyze and control risks [10]. While this works well with small projects [3,20]; software development domains that are highly regulated [15] such as automotive [11] and healthcare [12], as well as generally large projects [4], are bound to have higher number of iterations and some risks will be left unidentified or forgotten in the process. The resultant effect could be faced in the later stage of the development cycle and this will likely be costlier to rework [3,4].

To avoid this kind of scenario, it is important to incorporate risk management procedures explicitly with the agile methods [13,14,15,16]. This is necessary because risk management is a systematic process that is continuous all through the development of the software and in fact is a project of its own [17] that should be integrated formally. Managing risk in an unordered and unconscious manner will not deliver the best result. Agile methods do not recommend specific procedures to support risk management [18]. They think that the short iterations they utilize curtail risks however this is insufficient [19,20]. According to [21], the inherent risk management procedure by SCRUM which is an agile method is not as good as in the case of risk management in traditional software development methods because some steps of the risk assessment are not fulfilled except for the activities of the risk identification. Consequently, it is suggested that SCRUM be enriched with some selected steps from PRINCE2 risk management which promises a better result for delivery even in global software development projects [21]. According to [22], in most cases, agile methods are preferred to traditional methods as the features they possess bring about flexibility and swiftness in delivering the software product, but it also stressed that agile methods fail atimes [23] due to among others, the inadequate risk management [24]. This implies that lack of risk management is risk in itself [5]. Agile teams are democratic and self-organizing ideally, consisting of developers with all the necessary required skills, however, they do not consider a risk manager [2,3,25,26] whose sole duty is to take note of all risks identified by the team and also ensure that the risks are managed properly. It is believed that risk management is passive and implied in the agile process. [25,27] states that though Agile methods are speedy ways of developing software, the need to implement a robust risk management practice cannot be ruled out. Better ways to integrate proactive risk management measures need to be put in place carefully without compromising the agile spirit. It is categorically stated by [28] that Agile methods need a formal technique to manage risks when multiple agile teams work on the same product, stressing that a higher coordination effort is required and more formal practices are applied.

Even proponents of Agile Methods attest to the fact that former risk management is necessary for some projects. According to [28], most successful software projects utilize the hybrid of traditional risk management and the agile software cycle. In other words, traditional risk management and agile software cycle are complementary. Meanwhile, traditional risk management follow a formal process.

As a practitioner, he pointed out that in Agile methods, formal way of managing risk becomes necessary when the project is expensive, has many touchpoints, and involves the use of new technology. According to [29], high- risk projects built in the agile environment need risk management implemented explicitly. Furthermore, [30,31] state that risks and nonfunctional requirements (NFRs) are not well defined in agile settings. However, there exist nonfunctional requirements-related risks that can threaten software development [32]. This is obvious as agile methods primarily are tailored to delivering fast and efficient software products which happen to be the functional requirements aspect of the software. Proper consideration of Functional and nonfunctional requirements together brings about good software quality. Thus, inconsideration of the nonfunctional requirements brings about the poor quality of software architecture. Utilizing some external procedures in managing NFRs will greatly reduce the risks related to them. Previous works were made to identify possible risks that can emerge when utilizing agile methods in projects and in this paper, the researcher intends to review and deduce based on the risks identified, the relevance and benefits of formally managing risks in the agile development projects.

2. METHODOLOGY

A study of the SCRUM framework, a popular agile method is made, first of all, to explore development steps and features in agile methods generally. This was followed by the review of related research papers from peer-reviewed journals and other sources on software development projects implemented using agile methods without any form of formal risk management procedures. The aim was to ascertain the sufficiency of the inherent risk control measures in the agile setting and if not sufficient, extract risks identified in such projects. A summary of risks identified in the reviewed studies was made. Thereafter, the researcher was able to deduce the relevance and benefits of implementing explicit risk management in agile software development projects. The conclusion was then made and further studies suggested.

3. RISK MANAGEMENT IN A TYPICAL AGILE ENVIRONMENT

Scrum, Kanban, Xtreme Programming, Crystal, Lean and Feature-driven development are flavours of Agile development methodologies. To explain the procedures followed generally in an agile setting, scrum is illustrated here. From the studies reviewed, scrum happens to be a popular agile methodology [30,33,34] hence the reason for our choice.

Typically, In Scrum, there are mainly three roles namely the Product Owner, Scrum Master, and the team. The product owner is a representative of the owner and user of the software under development. In other words, the product owner is the project’s key stakeholder. The ScrumMaster ensures the team is as productive as possible. He guides the team and prevents deviation from the goal by removing impediments. The team is made up of 5 – 9 people nonetheless scrum can be used for bigger projects by utilizing many teams. A scrum team is made up of diverse sets of skills suitable enough to develop software products with little or no supervision. Thus, the scrum team is said to be cross-sectional and self-organizing.

At the start of the project, a project vision is made known, and features required in the product to be developed are listed in their order of priority by the product owner. This forms the product backlog. The sprint or iteration, which is the time set by the team to complete a selected list of features from the product backlog is set. This time-boxed sprint usually within 2 – 4 weeks becomes a fixed time for a sprint throughout the project. The selected features list from the product backlog is to be accomplished in a given sprint from the sprint backlog. The team organizes each feature in the sprint backlog into a task list and executes these tasks of each feature. Once started, no interruption by way of adding or removing from the sprint backlog is entertained until the end of the sprint. However, changes can be made in the product backlog. During the duration of each sprint, on daily basis, a stand-up meeting that lasts 15 minutes is made by the team where each member of the team states what he/she carried out the previous day, and what he/she plans to achieve that day, and whatever obstacle encountered.

At the end of each sprint, a sprint review meeting is held. The features built are implemented before the product owner to check if requirements/ features are met. Thereafter a

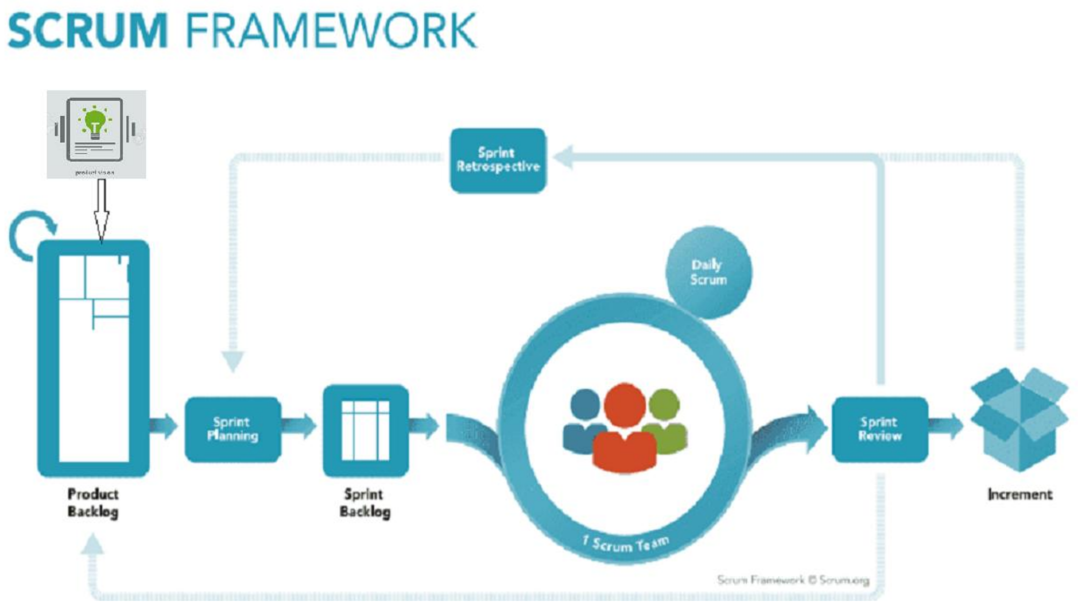

Fig. 1. SCRUM Framework (scrum.org)

sprint retrospective meeting is held by the team members to evaluate obstacles encountered in the just-concluded sprint and how to improve on them.

Another list of features from the top of the product backlog is moved to the sprint backlog for implementation in the next sprint. This is done until all features are completed. All other agile flavors follow the same steps but with slight modifications. The inherent practices in agile namely daily standup meetings, splitting implementable features into chunks, use of time-boxed iterations, review, and retrospective meetings all help to mitigate risks. However, from the literature reviewed these inbuilt measures of tackling risks are insufficient. The flow of activities in a typical scrum environment is shown in Fig. 1.

Moran A [35] listed business risks, financial risks, and technical risks as the major risks in software development. In his study, testing, validation, and documentation that ensure regular delivery in short increments as well as continuous integration practices are seen as being practiced in the scrum, a popular agile method. However, agile teams hardly do documentation. Moreover, in large projects, the number of iterations will increase and consequently, result in the omission of some risks untreated. The engagement of cross-functional teams is seen as a complete solution to improving the awareness and use of the latest technologies and ideas that will result in the production of quality software. This is true, however, the likelihood of leaving some risks unattended is obvious as there is no clear handler of risk management [4]. It is perceived that the product owner plans how the budget utilized will be in the project. He ensures that the expectations of the stakeholders are met at quicker releases thereby reducing the cost of production, but this is seeming in smaller projects where iterations and complexity of software are mild.

4. THE NEED FOR EXPLICIT RISK MANAGEMENT IN AN AGILE ENVIRONMENT

Derfer [36] study on risks in agile methods of development revealed that the inherent risk control practice utilized in agile is insufficient and suggested the use of a formal risk management approach. The study identified some risks which occurred as a result of the use of the agile method itself and other risks which become more pronounced when the agile method was introduced. The study was conducted in a large telecommunication-based company. Six projects which involved the customizing of a mega software product to suit the needs of customers in their respective operation locations were carried out. Major development and testing efforts were made. Customers' requirement needs and where the product is used were different though the same technology was deployed. Members of the development team working on a project are located in the same center however the project teams are located in different geographical areas. Three of the projects had their development teams in Poland, while two other projects had their teams in South Africa and the remaining one in Germany. Customers resided in locations other than the project development team that executed the project. Customers were large mobile network operators in Africa, Europe, and Latin. Scrum, Extreme programming, and Agile Project Management (APM) were the agile methods used in the six projects. Though in five of the projects, teams had no prior experience with agile methods, thorough training on agile practices was made. Scrum was the main method utilized though some practices of XP programming and Agile Project Management were applied. Semi-structured interviews were conducted with key project stakeholders. The interviews were aimed at identifying the strength and weaknesses of the examined projects as regards Agile methods as well as risks and opportunities for future Agile projects. Identified risks were as follows: risk of neglecting continuous integration, development process risks, development system risks, and contract risks, Most of these risks are further sub-divided into more risks. Development process risks were subdivided into Inefficient Scrum meetings and ineffective Scrum roles, Team not being able to self-organize and make group decisions, Wrong team decisions, and Misuse of self-organization to stop/revert the adoption of the Agile methodology. Development system risks were subcategorized into the lack of or limited compatibility of tools with Agile practices and missing infrastructure at the customer's site. They are risks that emanated from the introduction of Agile or became more visible when agile was introduced.

According to [37] the method of task prioritization as a way of managing risks in the agile methods is inadequate. He pointed out that such activities as the proper use of resources in achieving the enterprise goals are not inherent within the execution of tasks but are very important to be taken care of as they are considered part of the project. In other words, there are risks that cannot be addressed by the implicit way of development employed by the agile methods.

Elbanna A [38] stressed the need for implementing risk management in an explicit manner. According to him, techniques such as time-boxed iterations, demos, retrospectives and team ownership of each sprint's commitment go a long way in addressing some risks in the agile projects, however, these mechanisms are inadequate as they cannot be used to take care of risks in projects that are of medium to a larger size which are beyond a team's control. [38] emphasized that agile processes are better at identifying risks at the early stage of development rather than proffering solutions to identified risks.

Despite the fact that agile-based projects handle risks dynamically utilizing their inbuilt approach that is iterative and seasoned with daily meetings which together minimize risks, [39] explained that they still require some form of the explicit method of managing risks which is imperative but lacking. From the literature reviewed, proper risk management procedure is a necessity as its absence in a software project results in failure of varying magnitude. The study revealed that globally only 16.2% of software projects are completed within the stipulated budget and time while 52.7% and 31.1% of projects are faced with issues of late delivery and budget overrun and outright cancellation respectively.

The result of the study conducted by [40] confirmed that agile methods by their inherent practices mitigate certain risks which had helped to facilitate speedy means of completing software production. However, their studies also discovered some risk factors associated with agile software development methods. This implies the necessity of the use of other means of managing risks and also further buttressed the significance of utilizing a formal means of mitigating risks associated with agile methods. In their study, the Agile method of software development practices was examined in 28 organizations. 112 Interviews and 25 interviews were conducted in the 28 organizations and with other agile software development consultants and contractors respectively. This was to investigate the reasons behind the increase in failure and let-up in the use of agile software practices by some companies. Key risk factors related to Agile methods were identified. These include technical debt, fractured development and operation, increased defects in newly formed Agile Software Development (ASD) teams, Fragmentation of project management tools, and Knowledge Retention. Obviously, agile software developers are faced with conditions that sometimes threaten the smooth flow of the process, which is likely to compromise agility.

According to the study result of [2], an estimated savings of 40% is made when a formal risk management process is integrated with the agile software method in software projects. A mathematical risk management model for use in the agile methods was developed to implement an explicit risk management practice. Features that will estimate risk management process cost, as well as cost-benefit for implementing a formal risk management procedure in agile software development projects, were achieved. Use case model and activity diagram were developed to capture and understand activities involved, thereafter numerical calculation procedure derived appropriately, adopting risk cost estimation tool by [41]. Derivations for formal risk management cost and cost-benefit for implementing the formal risk management process made.

An online survey study that involved the distribution of questionnaires to 54 agile adopters by [42] was made. It was aimed at unearthing risks faced by agile practitioners and how such risks are moderated. Findings revealed that among others, requirements and schedule risks are the most common risks faced in software development projects using agile methods and except for the inbuilt risk control measures, there are no formal ways risk management procedures are followed. It was also revealed that agile software developers seldom assign risk management roles to keep track of identified risks and their management throughout the development process. This is evident in their result that over 80% of the respondents confirmed no risk management role was assigned in their project teams. It implies no record keeping of risk data and this, in turn, means some risks are likely to be half-handled and even overlooked. This could be the reason for most software project failures.

Holvitie J [43] Study on how to improve risk management in large agile settings was made. A case study in a moderately large eCommerce company was conducted by interviewing four production leads and four cross-team project managers. While each Production lead took supervision of a single team where features of the product are developed in an autonomous way, the Cross-team Project leads supervised multiple agile teams where more than one team works on a requirement (s). Findings revealed that whereas the implicit risk management procedures were adequate for the single teams who work independently on a particular feature, more formal risk control strategies are required for the latter (cross-teams) where many teams work on a particular feature. Reasons given for this conclusion are that there are no clear organized responsibilities laid down on who and how risks are to be managed and this indicates that explicit risk control measure is necessary for the production of large software. Consequently, the study recommended complementing agile practices with the Traditional software development practice in mega software projects.

A survey study by [44] compared risk management practices utilized in traditional methods and agile methods with the key aim of identifying risks associated with the methods. It reveals that Agile methods are suitable for short-term projects that require little planning and documentation as opposed to traditional methods. In addition, agile methods are flexible such that requirements are modified and or added at every stage of development as opposed to the rigidity applied in the traditional methods. However, it was deduced that the use of agile methods itself, can lead to project size creep, improper sprint planning, absence of specific experts can cripple the development process, users unable to assess software releases before the next release of another module, lack of progress tracking among developers in a complex environment.

5. SUMMARY OF EXISTING RISKS IN THE AGILE SOFTWARE DEVELOPMENT

Most of these risks are further sub-divided into more risks. However, sub risks are treated separately here.

5.1 The Risk of Neglecting Continuous Integration (CI)

Derfer [36], Institute for Agile Risk Management [14] Continuous Integration is a software engineering practice and in fact a best practice in agile software methodology. It involves routine integration of code changes into a shareable version control repository repeatedly and testing changes as early as possible to ensure defect-free product production. Failure to implement CI may result in spending more time on rework, uncertain project completion time, and more development effort.

5.2 Improper Sprint /Iteration Plans [44,22]

Sprint/Iteration plan is a plan done at the beginning of each sprint/iteration planning meeting to agree on and shortlist user stories or items the agile team can complete in the sprint and how work can be done. Thus, bad sprint plans can make the team set expectations that are unrealistic thus making the development process complex.

5.3 Risk of Team not being able to Self-organize and make Group Decisions [36,18]

Self-organization is an integral part of agile software development process. Though it can improve team performance when appropriately done, it also has risks associated with it. For instance, when the team has too much work to process, coupled with the involvement of a team member with narrow expertise and knowledge. In agile projects, apart from the team executing tasks in iteration builds, they also monitor, control, and make decisions on how the tasks are done. Thus, team members do not only do project execution tasks but managerial tasks as well. Wrong decisions are likely to be made if full knowledge about the project is unknown and when the decision of a team member is accepted because he/she is popular instead of on merit.

5.4 Missing Infrastructure at Customers' Site for Proper Customer Involvement in the Development Process [36,18]

All necessary infrastructure (both hardware and software) needs to be put in place. Such as dedicated hardware which includes hardware networking components and their installation. In fact, all required for implementing the test platform should be provided and put in place even before the commencement of the test otherwise the test will not be affected at all or done haphazardly, or even delay the agile spirit if it has to be arranged for test time.

5.5 The Risk of Lack of or Limited Compatibility of Tools with Agile Practices [36,18]

Agile engineering practices such as test-driven development and continuous integration require some tools different from those used in the traditional development scene. Thus, there are bound to be compatibility issues when such old tools are used in place of the agile-driven tools. In the existing literature, the risk of incompatibility was experienced. Besides, the high cost of running regression tests as well as the elongated run time of the testing process was evident.

5.6 Risks due to the use of Faulty Deliverables

This according to the study [36] involves the use of readymade components and software tools either produced in-house or external by the project team in the developing process to reduce costs and time without modification of such tools to suit its use in the agile settings.

5.7 Risk of Inefficient Scrum Meetings and Ineffective Scrum Roles [36]

Agile development process is associated with regular meetings as well as the introduction of new roles different from that in the traditional methods. The meetings consume substantial time and effort. New roles like the product owner and scrum masters introduced are costlier to maintain than the project roles of the traditional methods. Daily meetings and roles are vital agile practices, they do not add direct value to customers and are significant project overheads that if not managed properly will result in delays and overruns.

5.8 Project Size Creep [44]

The incremental feature utilized by agile allows for the addition of new features at each sprint/iteration of development. This could expand the project size and increase product cost.

5.9 Absence of Specific Experts [44]

Agile teams are cross-functional. This implies each team is made up of people with different expertise and skills. The benefit is that all capabilities required to undertake its scope end-to-end without help from another team are avoided. Supposing a team member leaves or dies, it could hinder the development process from achieving its goal at the specified time.

5.10 Users unable to assess Software Releases before the next Release of another Module [44]

Frequent software releases at the end of each sprint cycle prevent users from the privilege to completely assess previous releases as a result, key performance indicators are not properly spotted out.

5.11 Lack of Progress Tracking among Developers in Complex Environments [44]

This is the poor tracking, handling, and managing of risks, especially in large software projects [5,38,35,13,42,20,43]. In the medium to large projects, the number of iterations is large, and so also the risks that are identified. There must be a risk manager in each location. Depending on the size of the project and the number of locations, Risk manager(s) must be available in meetings to keep track of every identified risk. This can be effectively done by issuing risk track forms to team members who record risks identified. The risk manager enters these risks in an automated risk repository via risk application. That way risk details can easily be recalled and status updated. Such risk tools as Agile Risk Track Sheet (ARITS) and Repository as developed by [3] are suggested because they are web-based with smart database, and can be used by all teams working in different locations.

5.12 Lack of Control of Cross-teams Implementing a Single Feature [42]

The engagement of many teams in the implementation of a single feature of the software indicates that such a feature is large. This in turn means the software is a large one. According to [42,38], a formal risk management process is needed to exercise control over who will deal with which risk, keep records as well as track its migration.

5.13 Technical Debt [40]

Also known as tech debt or design debt or code debt) describes consequences that occur when development teams utilize easier and faster approaches (shortcuts) to meeting delivery deadlines usually a software functionality within the specified time [45,46]. These consequences in whichever form it comes namely code debt, test debt, documentation debt, and design debt involve some financial cost to refactor in the future. Thus, it is the result of prioritizing speedy delivery over perfect code [46]. Technical DEBT is noticeable in agile and in fact, accumulates over time due to their adherence to strict rules of delivering software features to clients within sprint in a consistent and continuous manner. Accumulation of such debt could lead to reworks and complexity which in turn could require more effort, money, project delay, and poor software quality as well in the long run [18].

5.14 Risks Related to Close Involvement of Business Stakeholders / Customers

According to the literature reviewed, though one of the successes achieved in agile is attributed to communicating constantly with stakeholders which makes them have an edge over the classical methods, stakeholders usually delist nonfunctional requirements such as security threats from the list of user stories. Probably due to a lack of understanding of the technicalities involved. Meanwhile, Security threats are evident and need to be checked continuously, failure to do so, further increases technical debt [40].

5.15 Fractured Development and Operation Risk [40]

According to [18], this refers to the disconnect between the agile development team (ADT) and the IT operations team. While the ADT focuses on building new software modules and applications as well as ensuring quick delivery to users, the Operation team ensures users get fast and bug-free software products that are stable and reliable in its operational environment. The IT operations teams usually include systems administrators, network engineers, and infrastructure specialists. On the other hand, a typical agile team consists of the product owner, team leader, specialists, architecture owner, development team members, scrum master, stakeholders which includes direct and indirect users, and, senior and portfolio managers [27]. Though both groups work towards the common goal of providing good software builds and services, their approaches in accomplishing that is different as they have different jobs, job priorities, work practices, and pace. As a result, the risks of delayed delivery of software builds within the scheduled time are evident. This is so as Operations Team attends to Infrastructure and service needs rather in a linear order contrary to the agile approach and pace of delivering builds in short time-boxed iterations. Another issue experienced in the research reviewed is that there is poor collaboration between the agile team and the IT operation team and this had resulted in the Agile team producing software not compatible for implementation in the operations environment without a rework. For example, reworking a software to suit its use in a server of the operating environment. The agile team has their sole interest in delivering working software within the allotted iteration without taking into consideration necessary knowledge of the operational environment and deployment such as

Infrastructure and networking operational issues which are necessary for the system to operate in its live environment, rework requests are not given the utmost priority that it deserves but developing of new features is at the foremost. Overall, the risk of delayed deployment of the developed software is likely to occur.

There are Increased defects in projects handled by Agile Software Development (ASD) teams that are new to an agile way of development [40]. This is risky as this implies more effort, time, and cost to correct such defects. It was however seen that as developers gain experience in the agile environment, the occurrence of such defects begins to decrease and reaches an acceptable level. This according to the study [40] was observed in small organizations.

5.16 Fragmentation of Project Management Tools [40]

Because Agile teams are self-organizing, they tend to choose project management tools they are conversant with to actualize their development needs. Though this motivates them and enhances their performance, it could also lead to confusion and complexities during the integration of work across domains and teams [40].

5.17 Knowledge Retention [40]

ASD teams by virtue of their policy, value face-to-face communication over written documentation. This means details of each system developed are retained until completion only if team members that started work on it are retained till the end and with the organization. Practically this is not so, members are often reshuffled and reassigned tasks to other teams from time to time. If a new member joins a team, velocity and quality drops. The reason behind this is that Light documentation is utilized in agile software methods. Thus, details about the system developed are not enough to expose the new team member to an understanding of the codes. Consequently, more time is used to unravel the details of the version of the software and the way forward.

5.18 Customers' Low Level of Interaction with Agile Team and in a Way Different from the Agile Way [36]

One major characteristic of the agile method of development is Customers' close interaction with the development team from the beginning to the end of the project. This is to ensure that software builds meet the requirement specification. However, according to the reviewed study, one of the risks noticed is that customers were not always in close contact with the development team and in most cases only make themselves available at the beginning and end of the project. With the manifestation of this risk comes another risk of customers not being available to give feedback and responses to the development team on requirements and other inquiries on the next iteration/sprint.

Requirements and schedule risks are the most common risks faced in software development projects using agile methods [36,41]. In an agile environment, users' needs can be modified at any time in the development cycle. This change if not well managed can lead to scope creep.

6. RELEVANCE AND BENEFITS OF EXPLICIT RISK MANAGEMENT PRACTICE IN AN AGILE SETTING AS DEDUCED FROM STUDIES REVIEWED

Generally, the relevance and benefits of explicit risk management are inherent in the various steps and roles of the risk management process.

Agile software development methodologies also manage risks but in an implicit way. The modular development of software and its unit testing in each iteration as well as integration testing with other modules alongside acceptance testing to check its conformance with requirements specifications are all effective ways to mitigate risks. In spite of this, from the literature reviewed, other risks do exist that occur with the introduction and use of the agile method itself. Also, there could be risks that surface when the project size exceeds a limit. Thus, managing risk explicitly will go a long way to address such risks. In a nutshell, explicit risk management in Agile software development methods will do the following:

1. Help to keep track of risk data by the introduction of a risk manager. Risks data are very important to keep track of risks identified, closely monitor the treatment of the risks identified and ensure their mitigation. Thus, the presence of a Risk Manager in the development team is very important to avoid skipping identified risks. Such a role is not explicitly spelled out in a core agile setting [3].

2. Help to initiate the proper use of resources in achieving the enterprise goals. Agile methods need to utilize some formal means of using relevant resources since it is not part of the execution of tasks assigned in the inherent process yet it is an important aspect [37].
3. Help to address risks that emanate from the introduction of Agile or became more visible when agile was introduced as revealed in the studies reviewed. Such risks can be well taken care of by external processes [36].
4. Help to manage risks when multiple agile teams work on the same product/feature. This is so because higher coordination effort is required, and the application of more formal practices is needed when cross-teams work on a product. According to [42], there are no clear organized responsibilities laid down on who and how risks are to be managed and this indicates that explicit risk control measure is necessary for the production of large software.
5. Generally, explicit risk management is a necessity in the development of large software projects using the agile method of development. The execution of large software projects in the agile environment is completed in many iterations and more efforts to track and control risks consciously is needed otherwise some risk may be left unattended to and may escalate to bigger problems at some point in the development cycle.
6. Explicit risk management may likely help to reduce monies expended for rework of risks by about 40% where only the inherent agile risk control practices are being utilized [41].
7. Explicit risk management will greatly reduce the risks that will emanate due to the improper management of nonfunctional requirements in agile settings. A good implementation of Functional and Non-functional requirements together results in the production of quality software [31,32]. Since agile methods are tailored to delivering software modules fast which happens to be the functional aspect of the development process, incorporating explicit means of managing risks associated with nonfunctional requirements is necessary.

7. CONCLUSION

Explicit risk management in medium to large scale agile software development projects is highly recommended to manage risks including those risks that will emanate as a result of adopting the agile method of development. Agile methods are methods practiced by many because of their features of swiftness in software product delivery however, the introduction of agile methods is associated with risks and must be dealt with. Large software projects are characterized by many features resulting in increase in the number of iterations. Without the deployment of the services of a risk handler whose sole duty is to take record of risks and track their mitigation, it will be difficult to identify risks and ensure the control of all the risks identified. There is no dedicated risk handler in a typical agile setting. Also, proper record keeping of risks details including what actions reduce or eliminate their effects will serve as historical records for such risks and will consequently guide developers on possible risks to expect and how to tackle such risks in similar projects. Again, though agile teams do some form of documentation they concentrate more on quick delivery of the software. Keeping track of risks details and their mitigation can be achieved by documentation or better still utilization of the Risk Repository system. Non-functional requirements [30,47] like security, and learnability are not taken care of using the existing inbuilt features in agile thus requiring external means of control. This is also an important aspect to achieve quality software. The cost of rework of identified risks possibly outweighs the cost of managing risks explicitly as seen in the literature reviewed and as such it is recommended that explicit risk management be incorporated into agile methods. In all, explicit risk management in agile methods of development is worth trying as it extends the utilization of agile methods to the development of all sizes of software development projects and highly regulated software projects. Further studies reviewing a larger number of agile software projects are suggested to help further explore explicit risks management benefits in agile software development projects. Though the findings used in this study included the review of completed agile software projects, ongoing agile software development projects would have been sought to identify risks condition and how the risks are possibly tracked or mitigated, or ignored. This can also be another point of study in the near future.

COMPETING INTERESTS

Author has declared that no competing interests exist.

REFERENCES

1. Olaronke I, Rhoda I, Ishaya G. An Appraisal of Software Requirement Prioritization Techniques. AJR CoS. 2018;1:1-16.
2. Thom-Manuel OM, Ugwu C, Onyejegbu N. An Extended Agile Software Development Project Budget Model.International Journal of Computer Science and Software Engineering. 2017: 6(12):306-314.
3. Thom-Manuel OM, Ugwu C, &OnyejegbuLN A New Mathematical Risk Management Model for Agile Software Development Methodologies. International Journal of Software Engineering & Applications. 2018; 9(2):67-86.
4. Cohn, Mike. Succeeding with Agile: Software Development Using Scrum. Kindle ed. Addison- Wesley; 2009.
5. Franck Marle. An Assistance to Project Risk Management Based on Complex Systems Theory and Agile Project Management. Complexity; 2020. Article ID 3739129, 20 pages, 2020. Available:https://doi.org/10.1155/2020/3739129
6. International Organization for Standardization. Risk Management. (ISO Standard No. 31000:2009).
7. International Organization for Standardization. Medical devices – Application of risk management to medical devices. (ISO Standard No. 14971:2019).
8. PMI - Project Management Institute. A Guide to the Project Management Body of Knowledge (PMBOK®Guide) Sixth Edition (6th ed.). PMI; 2017.
9. Hijazi H, Khdour T, Alarabeyyat AA. Review of risk management in different software development methodologies. International Journal of Computer Applications. 45(7):8-12.
10. Carvalho MM, Junior RR. Impact of risk management on project performance: The importance of soft skills. International Journal of Prod. Res. 2015;53(2):321-340.
11. VDA - Verband der Automo bilindustrie, QMC - Quality Management Center Working Group 13 - Automotive SIG. Automotive SPICE Process Assessment - Reference Mode – version 3.0; 2015.
12. MacMahon ST, McCaffery F, Keenan F. The MedITNet assessment framework: development and validation of a framework for improving risk management of medical IT networks. Journal of Software: Evolution and Process. 2016;28(9):817-834.
13. Hammad M, Inayat I. Integrating Risk Management in Scrum Framework. 2018 International Conference on Frontiers of Information Technology (FIT); 2018.
14. Institute for Agile Risk Management. Agile Risk Management; 2020. Available: https://agileriskmanagement.org/
15. Vieira, Marcel, Jean Carlo Rossa Hauck and Santiago Matalonga. How Explicit Risk Management is Being Integrated Into Agile Methods: Results From a Systematic Literature Mapping. 19th Brazilian Symposium on Software Quality; 2020.
16. Feizi Kamran, Asadi Gharabaghi Mehdi, Olfat Laya, Taghavifard Mohammad Taghi. Risk management in Agile Software Development projects: Designing a process model with a qualitative approach. Iranian Journal of Management Sciences; 2020. Cited 2022May02];15(57):1-27. Available:https://www.sid.ir/en/journal/ViewPaper.aspx?id=822394
17. Ylimannela, Ville. A model for risk management in agile software development; 2013.
18. Moran A. Agile risk management. In Agile Risk Management. Springer, Cham. 2014;33-60.
19. Chaoucha S, Mejrib A, Ghannouchia SA. A framework for risk management in Scrum development process. Procedia Computer Science. 2019;164:187-192.
20. Alharbi ET, Qureshi MRJ. Implementation of Risk Management with SCRUM to Achieve CMMI Requirements. Computer Network and Information Security. 2014; 11:20-25.
21. Katarína B, Šimíčková J. Risk management in traditional and agile project management. International Scientific Conference on Sustainable, Modern and Safe Transport (TRANSCOM). pp. 986–993 High Tatras, NovySmokovec – Grand Hotel Bellevue, Slovak Republic: Elsevier B.V.
22. Suryaatmaja K, Wibisono D, Ghazali A, Fitriati R. Uncovering the failure of Agile framework implementation using SSM-based action research. Palgrave Communications. 2020;6(1):1-18.
23. Dhir S, Kumar D, Singh VB. Success and Failure Factors that Impact on Project Implementation Using Agile Software Development Methodology. In: Hoda M,

Chauhan N, Quadri S, Srivastava P. (eds) Software Engineering. Advances in Intelligent Systems and Computing. 2019;731. Springer, Singapore. Available:https://doi.org/10.1007/978-981-10-8848-3_62
24. Micheal K. How is Risk Management in Agile Development Different From Risk Management in Waterfall Model?; 2012. Available:https://pm.stackexchange.com/questions/5957/how-is-risk-management-in-agile-development-different-from-risk-management-in-wa
25. Hammad M, Inayat I, Zahid M. Risk Management in Agile Software Development: A Survey. International Conference on Frontiers of Information Technology (FIT); 2019. Available:https://ieeexplore.ieee.org/abstract/document/8991647
26. Deloitte Future of risk Management in Financial Services: Integrating Risk Management and agile Projects; 2019. Available:https://www2.deloitte.com/content/dam/Deloitte/lu/Documents/risk/lu-risk-future-of-risk-series-integrating-risk-agile.pdf
27. Schön EM, Radtke D, Jordan C. Improving Risk Management in a Scaled Agile Environment. In: Stray V., Hoda R, Paasivaara M, Kruchten P. (eds) Agile Processes in Software Engineering and Extreme Programming. XP 2020. Lecture Notes in Business Information Processing, vol 383. Springer, Cham; 2020.
28. Wallmüller E. Business continuity. Chapter risk management for IT and software projects' (Springer-Verlag, New York, NY, USA,). 2002;165–178.
29. Ramos F, Costa A, Perkusich M, et al. A non-functional requirements recommendation system for scrum-based projects'. In 30th Int. Conf. Software Engineering & Knowledge Engineering (SEKE); 2018.
30. Iftikhar K, ALI S, Ngadi MDA. Enhancement of Non Functional Requirements in Agile Software Development. International Journal of Computer Science and Information Security (IJCSIS), 2016;14(12). Available on Enhancement of Non Functional Requirements in Agile Software Development | Journal of Computer Science IJCSIS - Academia.edu.
31. Matharu, Gurpreet, Mishra, Anju, Singh, Harmee, Upadhyay, Priyanka. Empirical Study of Agile Software Development Methodologies. ACM SIGSOFT Software Engineering Notes. 2015; 40:1-6. Available:10.1145/2693208.2693233
32. Kumar, Manish, Dwivedi RK. Applicability of Scrum Methods in Software Development Process; 2020. AvailableSSRN: https://ssrn.com/abstract=3610759 or http://dx.doi.org/10.2139/ssrn.3610759
33. Kendis Team. Risk Management in Agile Scrum; 2019. Available :https://medium.com/@media_75624/risk-management-in-agile-scrum-e0e5d18f0cf
34. Walczak W, Kuchta D. Risks characteristic to Agile project management methodologies and responses to them. Operaions Research and Decisions. 2013; 23:75-95.
35. Moran A. Agile risk management. In Agile Risk Management. Springer, Cham. 2014; 33-60. Available:https://scinapse.io/papers/1042857008
36. Derfer B. Introducing the Agile Risk Management Framework, Agile Six Applications, Inc; 2016. Available:https://www.agilegovleaders.org/wpcontent/uploads/2016/03/Agile_Risk_Management _ Framework.pdf
37. Prakash B, Vijay Viswanathan. Risk Prioritization for Software Development using Grey Wolf Optimization 1458; 2019.
38. Elbanna A, Sarker S. The Risks of Agile Software Development: Learning from Adopters. IEEE Software. 2016;33(5):72-79. DOI: 10.1109/MS.2015.150
39. Khatavakhotan AS, Hashemi NT and Ow SH. AMathematical Risk Management Model for Iterative IT Projects based on the Smart Database. International journal of information and Electronic Engineering. 2011;1(3):229-233.
40. Hammad M, Inayat I, Zahid M. Risk Management in Agile Software Development: A Survey. International Conference on Frontiers of Information Technology (FIT); 2019. Available:https://ieeexplore.ieee.org/abstract/document/8991647. doi:
41. Schön EM, Radtke D, Jordan C. Improving Risk Management in a Scaled Agile Environment. In: Stray V., Hoda R.,

Paasivaara M., Kruchten P. (eds) Agile Processes in Software Engineering and Extreme Programming. XP 2020. Lecture Notes in Business Information Processing, vol 383. Springer, Cham.

42. Alshathry S, Alnamlah B, Alkassim N, Jamail, NSM. Risk Management in Agile and Waterfall Models: A Review. International Journal of Advanced Science and Technology. 2020; 29(9):1149-1157.

43. Holvitie J, Licorish, SA, Spínola RO, Hyrynsalmi S, MacDonell SG, Mendes TG, Buchan, J, Leppänen V. Technical debt and agile software development practices and processes: An industry practitioner survey. Information and Software Technology. 2018;96:141-160.

44. Wolpers S. Technical Debt and Scrum: Who Is Responsible? Agile Zone; 2019.
Available :https://dzone.com/articles/technical-debt-amp-scrum-who-is-responsible

45. Productplan. Technical Debt; 2020.
Available:https://www.productplan.com/glossary/technical-debt/

46. Lynn R. Agile Roles in Software Development; 2020.
Available:https://www.planview.com/resources/articles/agile-roles-software-development/

47. Radigan D. Continuous Integration; 2020.
Available:https://www.atlassian.com/agile/software-development/continuous-integration

© 2022 Thom-Manuel; This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Peer-review history:
The peer review history for this paper can be accessed here:
https://www.sdiarticle5.com/review-history/84911