Association for Information Systems
AIS Electronic Library (AISeL)

MCIS 2014 Proceedings

Mediterranean Conference on Information Systems (MCIS)

Summer 9-4-2014

IDENTIFYING THE RISKS ASSOCIATED WITH AGILE SOFTWARE DEVELOPMENT: AN EMPIRICAL INVESTIGATION

Amany Elbanna
Royal Holloway University of London, UK,, amany.elbanna@RHUL.ac.uk

IDENTIFYING THE RISKS ASSOCIATED WITH AGILE SOFTWARE DEVELOPMENT: AN EMPIRICAL INVESTIGATION

Complete Research

Amany Elbanna, Royal Holloway University of London, UK, amany.elbanna@RHUL.ac.uk

Abstract

Agile software development has gained widespread acceptance and adoption by a broad range of organisations. Research to-date focuses on the positive gains from Agile adoption while the possible software risks have been largely overlooked. A recent failure case of a large Agile project invites a balanced view of Agile development and consideration of risks management. As a first step towards Agile risks management and in the absence of previous research on Agile risks, this research is exploratory and aims to identify the risks of Agile software development. To this end, qualitative data was gathered through 141 face-to-face interviews with Agile software developers, scrum masters, Agile project managers, CIOs and consultants. The analysis reveals surprising risk factors that are different from the traditional risk factors identified in traditional software development environment. To our knowledge, this study is the first that identifies the risks factors involved in Agile software development and we hope it paves the way for a more balanced view on Agile software development.

Keywords: Agile software development, software risk management, agile risks, software development risks.

1 Introduction

Agile software development concept and methods emerged out of the professional discontent with traditional approaches and high failure rate of software development projects together with the need to find faster ways to develop software to cope with fast moving business environment (Baskerville et al. 2003; Baskerville et al. 2004). They started as practice-led movement of a group of software developers and professionals who signed an 'Agile manifesto' in Snowbird, Utah[1] in 2001. Since then, they have been adopted in different organisational context and gained business popularity as 'faster, better, cheaper' approaches to systems development. The Agile Manifesto set up the focus of Agile methods by stating that they favour individuals and interaction, working software, customer collaboration, and responding to change over the traditional practices of focusing on processes and tools, comprehensive documentation, contract negotiation, and following of a plan (Fowler et al.

[1] More information on the Agile manifesto could be found at: http://agilemanifesto.org

2001). There are many agile methods however it is recognised that "all agile methods …conform to the tenets outlined in the agile manifesto" (Nerur et al. 2005).

Academic research on Agile software development, however lagged behind practice and maintained a weak theoretical grounding (Conboy et al. 2004; Vidgen et al. 2006), had also maintained a positive bias towards agile methods' adoption. Very few studies have highlighted challenges to migrate to agile software methods (Boehm et al. 2005; Nerur et al. 2005) and questioned its suitability for large software projects (Harris et al. 2009; Michaelson 2013) and its conflict with traditional funding processes (Cao et al. 2013). Within this positively biased professional and academic environment, the UK Department for Work and Pensions (DWP) has ceased in 2013 all agile software development on its originally planned £2.2bn Universal Credit program after the cost escalated to a possible £12.8bn, six times the originally planned figures (Ballard 2013). The case of Universal Credit presents one of the largest Agile Development projects and hence might be an exception case. The project might have also suffered from more fundamental problems associated with government's large IT projects. However, it draws the attention to the importance of considering the risks associated with Agile development and ways to manage them. Understanding the risks associated with Agile software development is important not only to balance the dominating positive view but more importantly to draw the attention to the unceasing importance of managing software risks. It also provide organisations with a more balanced view that supports their endeavours to build better software in a fast moving business environment. An important first step in managing risks is to identify and list risk factors (Schmidt et al. 2001). Hence, this research aims to contribute to this critical first step through answering the following questions: 1) what are Agile software risks as experienced by organisations? 2) why these risks occur?

Although prior research has identified software risks, IS project risks, and software project risks, no work -we are aware of- has examined the particular software risks associated with Agile software development methods. Hence the adoption of an empirical investigation of Agile software risks as experienced by organisations and participants of agile projects provides an appropriate research method for this research. A total of 141 face-to-face interviews were conducted with agile software developers, project managers, Scrum masters, CIOs, and IT operations staff who: 1) are involved or deal with Agile software projects at the time of data collection or recently completed one, 2) have completed at least one Agile project. The study identifies a set of novel and surprising Agile methods risk factors and provides insight into why these risks occur. In doing so, it contributes to advancing knowledge in Agile software development and support organisations in their journey to adopt agile practices.

The paper is organised as follows. The second section, after the introduction, provides a brief literature review on software development risk and agile software development methods. It reveals that while IS risk management presents a rich stream of IS research, investigating the possible risks of Agile software development did not receive any research attention. Section three briefly introduces the research framework and section four presents the research methods. Section five presents the findings and section six briefly discusses the findings and draws conclusions.

2 Literature review

2.1 Software risks

Risks are factors that can, when present, negatively affect a project unless project managers take appropriate counter measures (Wallace et al. 2004a). Considering the high failure rate of IS implementations and projects, IS research on risks regard IS risk management as a mean of improving the success rate of IS projects (Schmidt et al. 2001; Liu et al. 2010). It maintains that by identifying, analysing and controlling threats, action can be taken to reduce the chance of failure (Sherer et al. 2004a). IS risk studies varied in its focus from examining systems development risk in strategic context (Kemerer et al. 1991), IS project risks (Keil et al. 1998) including the risks of particular IS project such as ERP (Aloini et al. 2007) and outsourcing (Oh et al. 2006; Osei-Bryson et al. 2006), the risk of particular development mode such as end user computing (Alavi et al. 1986), software risks (Boehm 1991; Barki et al. 1993; Lyytinen et al. 1998; Na et al. 2007), and software project risks (Wallace et al. 2004b). A survey of research on IS risks found that "IS risk literature is a jumble of diverse models and partially overlapping, atheoretical lists of risk factors and risk components" (Alter et al. 2004, p. 3). This diverse focus of IS research on risks was also found to lack a practical model and provide a confusing agenda for managers to understand IS-related risks (Alter et al. 2004; Sherer et al. 2004b).

Research on IS risk management adopts one of four approaches namely; checklists, analytical frameworks, process models, and risk response strategies (Bannerman 2008). Checklists research aims to identify and rank risk factors and produce lists of highly ranked risk factors (Boehm 1991; Keil et al. 1998; Schmidt et al. 2001; Liu et al. 2010). These lists are usually compiled from surveys of the experiences of stakeholders who have been involved in software projects. The rationale behind this research is that risk factors experienced in a project may also be important in other projects. Hence, "Managers and system engineers can use the checklist on projects to help identify and resolve the most serious risk items…"(Boehm 1991, p. 35). Long lists of risk factors have been identified in the literature (Boehm 1991; Schmidt et al. 2001; Wallace et al. 2004a; Warkentin et al. 2009). This research was criticised for producing long checklists that are impractical and difficult for managers to use (Cule et al. 2000). Analytical framework research aims to reduce this complexity through consolidating risks into categories (Keil et al. 1998; Cule et al. 2000; Ropponen et al. 2000; Han et al. 2007) or adopting explanatory framework to synthesize risk factors and explain their socio-technical components (Lyytinen et al. 1996; Lyytinen et al. 1998; Carlo et al. 2004). Research that adopts process models tend to specify stepwise tasks and individual activities for managing risks. These process steps typically include risk strategy, risk identification, risk analysis, risk response and risk control (Bannerman 2008). Process research, however provides a practical guidance for risk management, lacks deep analytical underpinning to explain the occurrence of risks. Research on risk response strategies aims to describe generic options for responding to foreseen IS risks (Bannerman 2008).

An extensive review of IS research on risk is beyond the scope and limit of this paper and could be found at (Alter et al. 2004; Sherer et al. 2004a; Warkentin et al. 2009). However, despite the considerable number of studies on IS risk, their different focus and different

adopted approach, "one thing these authors have in common is their suggestion that if risk is to be managed, it must first be identified" (Cule et al. 2000, p.67).

The risks of IS in operation has received little attention (Rönnbäck et al. 2011). Out of 46 papers reviewed, Sherer and Alter (2004) found that most papers consider IS project risk and only 4 papers consider both IS project risk and risk of IS in operations (Sherer et al. 2004a). While risk factors overlap yet there are some unique risks to either software, projects or operations (Alter et al. 2004).

2.2 Agile software development

Agile software development methods continue to grow in popularity as organizations increasingly seek speed to market and ability to cope with changes in the current fast moving business environment. Agile methods present a practice-led movement to developing software in a high-change and high-speed environment (Highsmith 2013). Agile software development methods includes different strands or approaches such as *eXtreme Programming (XP)* (Beck 2000), *Scrum* (Schwaber et al. 2002), the *Dynamic Systems Development Method (DSDM)* (Stapleton 1997), *Crystal* (Cockburn 2001), *Agile Project Management (APM)* (Highsmith 2004), *Feature Driven Design* (Coad et al. 2002), and *Lean Software Development (LSD)* (Poppendieck 2001). However these approaches differ in their details, daily practices and emphasis, they share the Agile development values as presented in the Agile Manifesto (Nerur et al. 2005). They also share a general orientation towards short iterative development cycles, frequent communication with customers, and constant adaptation and accommodation of change. Abrahamsson et al (2003) state that "agile software development in general is characterized by the following attributes: incremental, cooperative, straightforward, and adaptive" (Abrahamsson et al. 2003). They explain that incremental refers to small software releases with rapid development cycles, cooperative refers to a close customer and developer interaction, straightforward refers to to the simplicity of the method itself and that it is easy to learn, adaptive refers to the ability to embrace and accommodate last moment changes (ibid). For the purpose of this research, we use Agile development methods as an umbrella term to reflect these common characteristics.

Following Agile methods, software development groups "concentrate only on the functions needed immediately, delivering them fast, collecting feedback and reacting rapidly to business and technology changes" (Abrahamsson et al. 2003, p.2). These work practices caught the attention of academic research to examine the differences between Agile practices and traditional approaches of software development (Baskerville et al. 2003). Research examined the characteristics of short cycle development (Baskerville et al. 2004), offered a broad conceptualization of agility based on manufacturing and systems thinking (Conboy et al. 2004) and a theoretical grounding based on complex adaptive systems (Meso et al. 2006). Research also examined the possible combination of different Agile methods (Fitzgerald et al. 2006) in addition to understanding Agile application in different contexts such as off-shoring (Sarker et al. 2009) and globally distributed projects (Cummings et al. 2009).

Advocates of Agile methods argue that Agile methods resolve many of the identified risks particularly on the interface between the software development and the business organisations. Factors such as poor communication with users and stakeholders, lack of users involvement, failure to manage end users expectations, misunderstanding of requirements and failure to accommodate change of

requirements and scope are argued to be positively managed through the adoption of Agile methods. Advocates of Agile methods also argue that the short iteration cycles allow for accommodating change and clarifying users requirements while the face-to-face interaction and constant communication with users and business stakeholders improves communication and develops common understanding of requirements (Elbanna et al. 2009). Table 1 exhibits some of the traditional software risk factors that Agile methods mitigate.

Few authors have discussed the challenges of migrating to Agile methods (Nerur et al. 2005) or hinted to the trade offs and balancing decisions between quality and speed (Pries-Heje et al. 2004) and the "trades of efficiency for speed and flexibility" (Highsmith 2013, p.12). Conducting a comparative analysis of nine Agile methods, Abrahamsson et al (2003) conclude that "none of the [agile] methods were either extensive or precise" and that "practitioners, currently, have partial solutions to problems that cover a wider area than the methods do" (Abrahamsson et al. 2003). Critics also highlighted "the potential for inefficient and chaotic software design and architecture that require re-factoring to be rationalized (Fowler 2000)" (Pries-Heje et al. 2004, p.13). These concerns and remarks draw the attention to possible risk areas. However, research on risk identification, assessment, and mitigation of Agile software development methods has been overlooked. This is surprising considering the longstanding research tradition on IS risk management.

Table 1: Traditional Risk factors and Agile methods impact

Traditional IT project risk factors (Schmidt et al. 2001)	Agile positive impact on traditional risk factors
Lack of top management commitment	• Positive impact has not been identified
Failure to gain user commitment	✓
Misunderstanding the requirements	✓
Lack of adequate user involvement	✓
Failure to manage end users expectations	✓
Changing scope/objectives	✓ (not seen as a risk)
Lack of required knowledge/skills in the project personnel	✓ (if staff could be retained within the same team)
Lack of frozen requirements	✓ Through short-iteration cycles
Introduction of new technology	✓
Insufficient/inappropriate staffing	• Positive impact has not been identified
Conflict between user departments	✓

3 Research Framework

The classic socio-technical Diamond model developed by Leavitt 's (1965) (Leavitt 1965) is one of the most influential conceptualisation in IS research (eg. Keen 1981; Sarker 2000; Heeks 2002; Lyytinen et al. 2008). It has been also applied to studies of software risks (Lyytinen et al. 1996; Lyytinen et al. 1998). This model depicts the organisation as comprising of four inter-related dimensions: Actors (people), Tasks, Technology, and structure. Figure 1 presents an adaptaion of this model to the software development.

Applying Leavitt's model to software risk, Lyytinen et al. (1996) developed a framework of software risk management. This framework provides a hierarchical view of software development. It maintains that software development takes place in three interrelated environments namely; the system environment, the development environment, and the management environment (Lyytinen et al. 1996). The system environment is the environment where the system will operate. The development environment is where software design and development takes place. The management environment shapes software management activities. These three environments are socio-technical, each comprises of the four inter-related dimensions of organisation as identified by Leavitt's model. Two related processes exist within his framework: the development process and the software management process. The development process, is a first order process and is generated within the development environment with the purpose of inquiring "into the system environment, anticipate effective ways to use software and thereafter implement the software system". The software management process is generated within the management environment and forms a second order process that manages the software process and its environment.

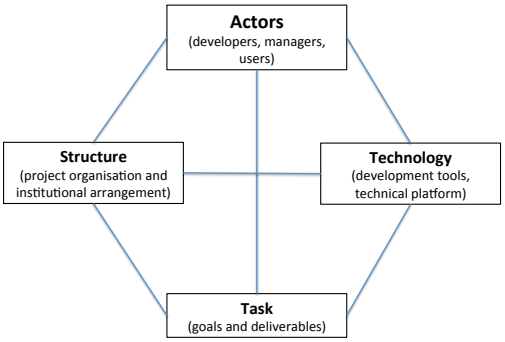

Figure 1: A Socio-Technical Model of System Development (Adapted from Lyytinen et al 1998)

According to Lyytinen et al. (1996) framework, software risks are borne within: 1) the system environment; 2) the development environment; 3) the management environment. This framework is adopted and adapted - as depicted in figure 2- in this study as a synthesising and analytical framework through which the emerging risk categories and their interrelationship could be interpreted.

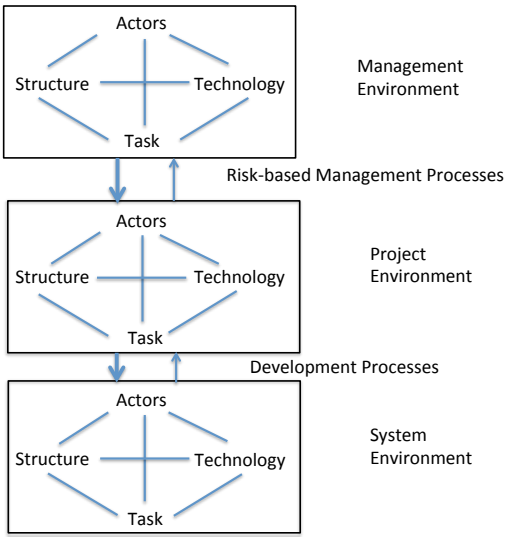

Figure 2: A framework for software risk management (Adapted from Lyytinen et al. 1996)

4 Research Methods

4.1 Data collection

This study presents an on-going exploratory and inductive research project on Agile software development adoption and the associated risks. It is inductive in the sense that findings are grounded in empirical data where no hypotheses or theoretical frameworks were adopted prior to data collection. It adopts a qualitative approach for inquiry (Garcia et al. 1997; Trauth 1997; Myers et al. 2002). Face-to-face interviews were conducted in organisational premises and in most cases in an Agile's project room or floor where velocity charts and white boards (if used) were posted on the wall. Interviews consist of 112 interviews in twenty-eight different organizations; four of which were examined longitudinally over a period of time that varies from 2 to 5 years. Previous research on IS risks showed that stakeholder groups tend to identify and highly consider risks that is lying outside their own domain and area of responsibility (Schmidt et al. 2001; Bannerman 2008). Therefore, a range of Interviewees from different stakeholder groups was included in the study. Interviewees include members of development teams and operations teams, Scrum masters, project management, IT directors, business managers and users. The studied companies are in the following sectors: financial services, utility, media and broadcasting, railways and automotive, professional services, e-commerce, and public services. In addition, 16 freelance developers and 8 consultants working on or leading 8 different Agile consulting firms were also interviewed. Early interviews were largely open-ended unstructured while later interviews tended to be semi-structured. Interviews lasted between 40-90 minutes. Documents such as Agile projects' wikis, Agile project management software tools and Agile projects' documents were reviewed in some of the visited organisations. In addition, nine Agile business-orientated conferences and workshops were attended and observed and informal conversations and discussions with participants took place. Table 2 presents a summary of data collection sources.

Table 2: Data collection summary

	No. of Observed conferences	No. of interviews	No. of organisations	sector of companies
Stakeholders Interviews: people/organisations		112	28	Financial services, utility, media and broadcasting, railways and automotive, professional services, e-commerce, and public services.
Consultants interviews: People/organisation		24	8	Agile consulting firms
Conferences' observation	9			
Confirmatory interviews		5	2 + (2) previously visited	
Total	9	141	37	

4.2 Data analysis

To identify risks, we followed the Condition-Transition-Consequence (CTC) framework of risk as identified by Gluch (1994). The Condition-Transition-Consequence (CTC) framework for software risk maintains that to identify a risk it is required to identify the conditions coupled with the concern about the potential consequence while the details of the transition and the process "need not be specified in the identification process" (Gluch 1994, p.13). This framework is presented as follows.

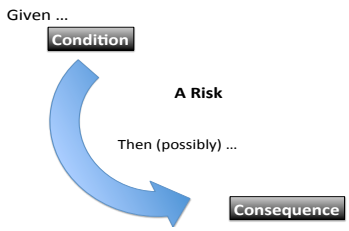

Figure 1: The CTC framework of risks identification (adapted from Gluch, 1994)

Data analysis followed the convention of qualitative data analysis (Miles et al. 2014). First, all interviews were read carefully. Second, all interviews were open coded to discover negative incidents and risk factors. A list of risk factors emerged from this initial data analysis. Third, a second order coding was used to group similar risk factors into categories according to origin. Lyytinen et al (1996) framework was found to well resonate with the emerging categories, it was hence adopted to provide an overarching analytical framework. Fourth, these categories were further analysed following Leavitt's (1965) identified four dimensions which is part of Lyytinen et al (1996) framework. As the results were surprising to the researcher and has not been previously identified in the literature, five confirmatory interviews were conducted with five participants; two from two previously visited companies and three from two companies that were not included in previous stages of data collection. As it is an on-going research effort, further confirmatory interviews are also planned. To increase the validity of the findings, the results were presented in two business-orientated workshops on Agile software development.

5 Analysis of Findings

This section presents the research findings. Following Lyytinen et al. (1996) framework of software risks (figure 2), the research findings are categorised according to the three identified environments namely; system environment, development environment, and management environment as follows.

5.1 Systems Environment: Fracturing the development and operation

The development process as identified by Lyytinen et al (1996) brings together the development environment where the software is being designed and developed with the system environment where the software will be implemented and operate. Data analysis revealed that while Agile software development brings the business side and IS development side closely together in terms of people and tasks, it distanced the Agile development teams from the IT operations teams. It highlights possible misfit between the system environment and the development environment where the following risks surfaced. The system environment refers to IT operations and includes the systems, work practices and different administrators and infrastructure specialists that ensure the stability and reliability of software in its live operations.

5.1.1 Gap between the development and operations structures

Development teams that follow Agile methods work in time-boxes of 1 to 3 weeks depending on the team. This rhythm of work contrasts with the rhythm of work of operation teams. Operations teams work is rather linear fashion based on logging in requests and responding to them in queues. This misfit in the work rhythm caused delay in detecting network and deployments problems. It also created tension in the relationship between development and operation teams. On one hand, Agile teams found operations teams' response to requests be "too slow" and typically respond outside the iterations timeframe. On the other hand, operations teams found Agile teams' continuous requests to be "stressful" and jamming to their work sequence and provisioning of applications and infrastructure tasks.

5.1.2 Gap between the development and operations technology

Operations teams (Actor) are not part of Agile team structure and composition in the studied organisations. In addition, their work pace was found to be different (linear based on queues) than Agile teams. Together, this caused lack of communication and involvement of operations in the development and the risk of Agile teams missing credible accounting of the operations environment. As Agile teams work under the time pressure of short iterations, they focus on the development tasks, business requirements and desired business functionality. Hence testing such as unit testing, test-driven delivery, and User Acceptance Testing (UAT) were geared to ensure the standard of functional requirements and general performance of the software. Development tasks take place in an environment that could be different from the production or live operation environment. For example, in one of the studied organizations, the development was done using a version of a particular database technology that was different from the one used in the production environment causing deployment delay where considerable re-work had to be done to prepare the application for the existing operating environment. As technology rapidly advances, developers did not necessarily have the required

operational knowledge and experience which caused delay in deployment and production. A CIO explained: "I am afraid that the time we save in development is actually wasted in operations".

5.1.3 Gap in technology - structure in the development environment

The increasing use of off-the-shelf components particularly in web development requires specialists to operate them. Web development includes components such as web servers, application containers, networking, databases that are not integrated into the software yet they need considerable effort to configure and maintain. These tasks were seen to be beyond software developers' domain and capability and hence not included in Agile backlogs.

In non-web development, the focus on delivering user stories and business-led priorities leaves infrastructure issues such as servers or network upgrade out of the scope of iteration cycles till towards the end of the project close to going live stage causing delay. As one of the respondents eloquently put it "they keep discovering things [such as] we need a new server or network upgrade but very late in the development, these things should have been planned for very early on. You don't build and wait or build but cannot push to live environment".

5.1.4 Gap between task - structure of development and task - structure of operations

Defects reports and enhancements requests from operations for features that have been accepted by the business tended to be postponed by Agile teams as they focus on business requirements and delivering features for customers. This was particularly the case in developing for internal customers. These delays caused emergency patches and hence additional work for both Agile and operations teams. For example, in one of the studied organisation, an operations team's member expressed his concerns that the way Agile teams work had increased his and his colleagues "firefighting" in their attempt to make systems stable and available.

5.2 Development Environment

5.2.1 Technical debt: task-structure gap

Technical debt is the most reported risk between development team members. Technical debt is a term used by many of the respondents to refer to the lack of attention agile teams pay to code clean up, database problems, communication protocols and other non-functional requirements. Whenever not using this term, interviewees referred to it as problems and issues regarding addressing non-functional requirements. Incurring technical debt brings short-term benefits such as reduced development time or effort. However, it could result in unexpected project delays, and lower software quality in the long run as the software becomes more complex, less understandable and more difficult to maintain. Previous research also highlighted similar downside or cost of technical debt (Guo et al. 2011; Lim et al. 2012). It can also increase the risks of security breaches and data corruption. Participants knew that many of the technical debts have to be dealt with in the future and described that the pressure of delivering in short time-boxed iteration cycles made them take "shortcuts" to speed up the delivery. They also highlighted that business participants in Agile projects in most cases do not want to hear "technical jargon" and do not pay attention to its importance during the project as long as they are

invisible and do not impact immediate performance. Therefore, logging in technical and non-functional requirements in the iteration cycles is not always appreciated by the business and not part of business priorities hence tend to be pushed back in the delivery cycles. A technical lead of a development team expressed it as: "we have to tell them [the business customer], you know what, if you do not give us time to tidy up, the site will be slow, it is working fine now but I cannot guarantee this unless we fix it properly, only then they will give us an iteration… yes I have to flag out the threat". A product owner expressed another solution "over Christmas, the business is quite and we seize this to clear some of our technical debt without them [business] noticing".

5.2.2 Task-structure gap in less experienced teams

In smaller organizations and in particular in early adoption projects, interviewees stated that the number of defects have risen in their early Agile projects particularly the first one. However, it fell to its usual level in the following projects. Respondents found the experience and quality of developers to be more important in Agile development than traditional methods. Interviewees agreed that the quality of the product in Agile development depends largely on the quality of developers and their experience. Novice developers who lack previous Agile development experience tend to perform poorly in their first Agile projects. Reflecting on her early days in an Agile development team, a developer expressed the frustration she once felt when started Agile development saying: "when I came here, they were doing Agile, I found myself spending my time talking and not actually doing it...".

5.2.3 Gap in Actors-task: retainment

Agile development focuses on face-to-face communication over written documentation. This implicitly assumes that members of the same team could be retained till the end of the project and members of the development team of a particular system (application) could be retained in the organization to oversee the development of the following versions of the systems (application). In practice, these assumptions are hard to materialize. In a multi-project environment, the shuffling and re-assignment of team members is common practice. Also the high turnover in IT personnel means that it is unlikely that all the development team would be still working in the organization to oversee the following versions of the application or continue the work in a particular project. With light documentation and not having the same team that developed the application before, the following versions of the application tend to take longer time as the new team tries to unravel the logic behind the code and understand why particular development decisions were made. Also Agile team velocity and quality tend to drop with the joining of a new team member.

5.2.4 Gap in Actors-Task: business Involvement

Agile teams work through iterations where user stories are detailed and clarified at the beginning of the iteration where they are selected for development. The assumption is that business participants are committed throughout the development effort and hence they detail users stories at the start of the iteration and they will attend the end of iteration demos and participate in the iteration retrospective to learn from the experience. In practice, business involvement varies from one financial period to another and from one team to another. At business peak times, the involvement of the business significantly drops as business priorities change to attending to other business immediate needs. Also in professional services firms such as law firms and consultants where the time of business

participants is very costly, business participation tend to be sporadic and minimal. In the absence of business participants, development teams tend to assume (guess) the details of the available users stories. This might not be what the business actually needs and when business participants eventually attend, they would require amendments to these features. This is further amplified when the business participants do not also attend the end of iteration demos as unrequired features could continue to be build on and when discovered could cause significant delays to amend. The fluctuation of business involvement result in some iterations to be more focused with clear user stories and hence more productive than others. which could paralyze the work of Agile teams.

5.3 Management Environment

5.3.1 Gap between Task-structure of management environment and development

In large companies, project approval process requires up-front requirements to be reviewed by a board or committee that conduct strategic review for new IT initiatives and approve it if in alignment with business strategy. Also in highly regulated business, such as financial services companies, the proposed system has to go through another board to ensure its compliance with regulations and legal obligations. These corporate-required documents are typically prepared by different parties in the IT department. Hence once approved, an Agile team gets formulated. The Agile team will face the difficulty of dealing with already 3-6 months of work that they were not involved in. This causes "upfront bottleneck" as the team tries to understand and unravel the requirements and try to move from the documents to agile mode of working. A similar bottleneck occurs at the end of the development in highly regulated environment when institutional quality assurance and accreditation process are involved. A recent study on information assurance and Agile at the American Department of Defence (DoD) had also found similar results regarding the latter factor (Bellomo et al. 2012).

5.3.2 Fragmentation of software management tools

As Agile methods advocate self-managed teams, in practice Agile teams work with considerable freedom and autonomy. In their attempts to self-manage and develop their own work routine, agile teams select and use their own project management and testing tools. For example, in most of the studied organizations, Agile teams had moved away from the traditional white board and sticky notes that Agile practices were once known for. Many developers find white boards and sticky notes rather inconvenient as notes tend to fall down and lose their order. Moving away from the past concern of 'which sticky note is the stickiest' (Project Management Stack Exchange 2011), they use Agile project management software to organize their effort. The choice of this software largely depends on developers' knowledge and preferences. In many organizations, this led to the use of several different software to manage Agile projects. This is particularly the case as teams adopt cloud-based solutions such as Jira, Trello, Visual Studio, Asana and a variety of other tools including Google docs and Share Point. IT mangers find merging teams and managing workloads challenging under this condition.

Another example is the use of version control systems and User Acceptance Testing. In one of the studied organizations agile teams used 8 different version control systems and 4 UAT protocols. Being autonomous, Agile teams hold the view that "I use what I am comfortable with and they [other teams] can use whatever they want and happy with". Besides losing the organization the negotiation power and economy of scale in licensing, this fragmentation in tools negatively impacts knowledge sharing across projects and teams. It also confuses the operational teams who find it difficult to find systems-related data.

6 Discussion and Conclusion

The findings of this study suggest that Agile development is changing the nature of IS project risks. It suggests that while Agile development tackles some of the traditional risk factors such as lack of user involvement, misunderstanding of requirement, failure to gain users commitment, and change of scope (Keil et al. 2002; Baskerville et al. 2003), it also introduces new risks.

Agile development is customer-centric. It puts the business on the centre of the development effort as development team work closely with customers. Both parties are involved in stand up meeting, prioritization exercise, elicitation of users' stories, and end of iteration demos. The rationale behind this close relationship is to: 1) cut down the waste in system's features as only the business-required and prioritized features will be developed, 2) improve the response to business requirements change, 3) improve speed of delivery and 4) bring business value with every iteration cycle. The focus on business requirements and quest for speed of delivering business-valued features carry risks on three inter-related dimensions namely; the system environment, development environment, and management environment. The risks emerge out of misfit between the components of a particular environment or the misfit between component of a particular environment and the components of other environments. Fracturing the relationship between software development and IT operations and the accumulation of technical debt emerged as important risk factors associated with Agile software development. These and other identified risks should be assessed and managed throughout Agile development projects to ensure successful delivery and operations of the developed software.

This research offers a fresh view of Agile software development. While previous studies examined the adoption of Agile methods and benefits gained, this research follows the tradition of studying IS risk management and examines the risks of Agile software development. In doing so, it is hoped to open up a debate on the nearly taken for granted view of the absolute advantages of Agile development and provide organizations with a word of caution when adopting Agile that considers risk management. Organizations are encouraged to assess and manage the risks identified and described in this study.

References

Abrahamsson, P., Warsta, J., Sipponen, M. T., and Ronkainen, J. Year. "New Directions on Agile Methods: A comparative analysis," 25th Conference on Software Engineering, IEEE Computer Society Press, Los Alamitos, CA, 2003, pp. 244-254.

Alavi, M., and Weiss, I. R. 1986. "Managing the Risks Associated with End-User Computing," *Journal of Management Information Systems* (2:3), pp 5-20.

Aloini, D., Dulmin, R., and Mininno, V. 2007. "Risk management in ERP project introduction: Review of the literature," *Information & Management* (44:6), pp 547-567.

Alter, S., and Sherer, S. A. 2004. "A GENERAL, BUT READILY ADAPTABLE MODEL OF INFORMATION SYSTEM RISK," *Communications of the Association for Information Systems* (14).

Ballard, M. 2013. "Why agile development failed for Universal Credit," *Computer Weekly* (4 July 2013), pp http://www.computerweekly.com/news/2240187478/Why-agile-development-failed-for-Universal-Credit.

Bannerman, P. L. 2008. "Risk and risk management in software projects: A reassessment," *Journal of Systems and Software* (81:12), pp 2118-2133.

Barki, H., Rivard, S., and Talbot, J. 1993. "Toward an assessment of software development risk," *Journal of management information systems*), pp 203-225.

Baskerville, R., and Pries-Heje, H. 2004. "Short Cycle Time Systems Development," *Information Systems Journal* (14), pp 237-264.

Baskerville, R., Ramesh, B., Levine, L., Pries-Heje, J., and Slaughter, S. 2003. "Is internet-speed software development different?," *IEEE software* (20:6), pp 70-77.

Beck, K. 2000. *Extreme Programming Explained: Embrace Change*, (Addison-Wesley: Reading, Mass.

Bellomo, S., and Woody, C. 2012. "DoD Information Assurance and Agile: Challeneges and Recommendations Gathered through interviews with Agile program managers and DoD Accreditation Reviewers," Software Engineering Institute, Carnegie Melon.

Boehm, B., and Turner, R. 2005. "Management challenges to implementing agile processes in traditional development organizations," *Software, ieee* (22:5), pp 30-39.

Boehm, B. W. 1991. "Software risk management: principles and practices," *Software, IEEE* (8:1), pp 32-41.

Cao, L., Mohan, K., Ramesh, B., and Sarkar, S. 2013. "Adapting funding processes for agile IT projects: an empirical investigation," *European Journal of Information Systems* (22:2), pp 191-205.

Carlo, J. L., Lyytinen, K., and Boland, R. J. 2004. "Systemic risk, IT artifacts, and high reliability organizations: A case of constructing a radical architecture," in *Sprouts: Working Papers on Information Systems, 4(4)*. http://sprouts.aisnet.org/4-4: Case Western Reserve University, USA.

Coad, P., and Palmer, S. 2002. *Feature-Driven Development*, (Prentice Hall: Englewood Cliffs, NJ.

Cockburn, A. 2001. *Crystal Clear: A human-powered software development methodology for small teams*, (Addison-Wesley: Reading, MA.

Conboy, K., and Fitzgerald, B. 2004. "Toward a Conceptual Framework of Agile Methods: A study of agility in different discipline," in *ACM workshop on Interdisciplinary Software Engineering Research*: CA, USA.

Cule, P., Schmidt, R., Lyytinen, K., and Keil, M. 2000. "Strategies for heading off IS project failure," *Information Systems Management* (17:2), pp 65-73.

Cummings, J. N., Espinosa, J. A., and Pickering, C. K. 2009. "Crossing spatial and temporal boundaries in globally distributed projects: A relational model of coordination delay," *Information Systems Research* (20:3), pp 420-439.

Elbanna, A. R., and Murray, D. 2009. "Organizing IS projects for innovation: A collective mindfulness perspective," in *Americas Conference on Information Systems (AMCIS 2009)*: San Francisco

Fitzgerald, B., Hartnett, G., and Conboy, K. 2006. "Customising Agile Methods to Software Practices at Intel Shannon," *European Journal of Information Systems* (15), pp 200-213.

Fowler, M., and Highsmith, J. 2001. "The Agile Manifesto," *Software Development*:August), pp pp. 28-32.

Garcia, L., and Quek, F. 1997. "Qualitative research in information systems: time to be subjective?," in *Information systems and qualitative research*, Springer, pp. 444-465.

Gluch, D. 1994. "A construct for describing software development risks," Software Engineering Institute, Carnegie Mellon University Pittsburgh, Pennsylvania 15213.

Guo, Y., Seaman, C., Gomes, Rebeka, Cavalcanti, A., Tonin, G., Da Silva, F. Q. B., Santos, A. L., and Siebra, C. Year. "Tracking Technical Debt: An exploratory case study," 27th IEEE International Conference on Software Maintenance (ICSM)2011.

Han, W.-M., and Huang, S.-J. 2007. "An empirical analysis of risk components and performance on software projects," *Journal of Systems and Software* (80:1), pp 42-50.

Harris, M. L., Collins, R. W., and Hevner, A. R. 2009. "Control of flexible software development under uncertainty," *Information Systems Research* (20:3), pp 400-419.

Heeks, R. 2002. "Information systems and developing countries: Failure, success, and local improvisations," *The information society* (18:2), pp 101-112.

Highsmith, J. 2004. *Agile Project Management*, (Addison-Wesley: Boston, MA.

Highsmith, J. 2013. *Adaptive software development: a collaborative approach to managing complex systems*, (Addison-Wesley.

Keen, P. G. 1981. "Information systems and organizational change," *Communications of the ACM* (24:1), pp 24-33.

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

Keil, M., Tiwana, A., and Bush, A. A. 2002. "Reconciling User and Project Manager Perceptions of IT Project Risk: A Delphi Study," *Information Systems Journal* (12), pp 103-119.

Kemerer, C. F., and Sosa, G. L. 1991. "Systems development risks in strategic information systems," *Information and Software Technology* (33:3), pp 212-223.

Leavitt, H. J. 1965. "Applied Organisational Change in Industry: Structural, technological, and humanistic approaches," in *Handbook of Organisations,* J. March (ed.), Rand McNally & Co.: Chicago, pp. 114-1170.

Lim, E., Taksande, N., and seaman, C. 2012. "A Balancing Act: What software practitioners have to say about technical debt," *IEEE Software* (November/December), pp 22-27.

Liu, S., Zhang, J., Keil, M., and Chen, T. 2010. "Comparing senior executive and project manager perceptions of IT project risk: a Chinese Delphi study," *Information Systems Journal* (20:4), pp 319-355.

Lyytinen, K., Mathiassen, L., and Ropponen, J. 1996. "A framework for software risk management," *Journal of Information Technology* (11:4), pp 275-285.

Lyytinen, K., Mathiassen, L., and Ropponen, J. 1998. "Attention shaping and software risk—a categorical analysis of four classical risk management approaches," *Information Systems Research* (9:3), pp 233-255.

Lyytinen, K., and Newman, M. 2008. "Explaining information systems change: a punctuated socio-technical change model," *European Journal of Information Systems* (17:6), pp 589-613.

Meso, P., and Jain, R. 2006. "Agile Software Development: Adaptive Systems Principles and Best Practices," *Information Systems Management* (23:3), pp 19-30.

Michaelson, R. 2013. "Is Agile the Answer? The Case of UK Universal Credit," in *Grand Successes and Failures in IT. Public and Private Sectors*, Springer, pp. 295-309.

Miles, M., Huberman, A., and Saldana, J. 2014. *Qualitative Data Analysis: A Methods Sourcebook*, (3rd ed.) Sage publications.

Myers, M. D., and Avison, D. 2002. *Qualitative research in information systems: a reader*, (Sage.

Na, K.-S., Simpson, J. T., Li, X., Singh, T., and Kim, K.-Y. 2007. "Software development risk and project performance measurement: Evidence in Korea," *Journal of Systems and Software* (80:4), pp 596-605.

Nerur, S., Mahapatra, R., and Mangalaraj, G. 2005. "Challenges of migrating to agile methodologies," *Communications of the ACM* (48:5), pp 72-78.

Oh, W., Gallivan, M. J., and Kim, J. W. 2006. "The market's perception of the transactional risks of information technology outsourcing announcements," *Journal of Management Information Systems* (22:4), pp 271-303.

Osei-Bryson, K.-M., and Ngwenyama, O. K. 2006. "Managing risks in information systems outsourcing: An approach to analyzing outsourcing risks and structuring incentive contracts," *European Journal of Operational Research* (174:1), pp 245-264.

Poppendieck, M. 2001. "Lean Programming," *Software Development Magazine* (9:5), pp 71-75.

Pries-Heje, J., Baskerville, R. L., Levine, L., and Ramesh, B. 2004. "The high speed balancing game: How software companies cope with internet speed," *Scandinavian Journal of Information Systems* (16:1), p 1.

Project Management Stack Exchange 2011. "Which sticky note is the stickiest?," 25 August 2011.

Rönnbäck, L., and Holmström, J. 2011. "A Case against the Checklist Approach: Exploring Epistemic Strategies in IT Risk Management," in *Nordic Academy of Management Conference*: School of Business, Stockholm University, Stockholm.

Ropponen, J., and Lyytinen, K. 2000. "Components of software development risk: how to address them? A project manager survey," *Software Engineering, IEEE Transactions on* (26:2), pp 98-112.

Sarker, S. 2000. "Toward a methodology for managing information systems implementation: A social constructivist perspective," *Informing Science* (3:4), pp 195-206.

Sarker, S., and Sarker, S. 2009. "Exploring agility in distributed information systems development teams: an interpretive study in an offshoring context," *Information Systems Research* (20:3), pp 440-461.

Schmidt, R., Lyytinen, K., Keil, M., and Cule, P. 2001. "Identifying software project risks: an international Delphi study," *Journal of management information systems* (17:4), pp 5-36.

Schwaber, K., and Beedle, M. 2002. *Agile Software Development with Scrum.*, (Prentice-Hall: Upper Saddle River, NJ.

Sherer, S. A., and Alter, S. 2004a. "Information system risks and risk factors: are they mostly about information systems?," *Communications of the Association for Information Systems* (14).

Sherer, S. A., and Alter, S. 2004b. "Information Systems Risks and Risk Factors: Are They Mostly About Information Systems?," *Communications of the Association for Information Systems* (14:1), p 2.

Stapleton, J. 1997. *DSDM: Dynamic Systems Development Method.*, (Addison Wesley: Harlow, England.

Trauth, E. 1997. "Achieving the research goal with qualitative methods: lessons learned along the way," in *Information systems and qualitative research*, Springer, pp. 225-245.

Vidgen, R., and Wang, X. 2006. "Organizing for Agility: A complex adaptive systems perspective on agile software development," in *Fourteenth European Conference of Information Systems,* J. Ljunberg and M. Andersson (eds.): Goteborg, Sweden.

Wallace, L., and Keil, M. 2004a. "Software project risks and their effect on outcomes," *Communications of the ACM* (47:4), pp 68-73.

Wallace, L., Keil, M., and Rai, A. 2004b. "Understanding software project risk: a cluster analysis," *Information & Management* (42:1), pp 115-125.

Warkentin, M., Moore, R. S., Bekkering, E., and Johnston, A. C. 2009. "Analysis of systems development project risks: An integrative framework," *ACM SIGMIS Database* (40:2), pp 8-27.