Sources of Interruptions Experienced During a Scrum Sprint

Maureen Tanner and Angela Mackinnon
University of Cape Town, South Africa
mc.tanner@uct.ac.za
MCKANG008@myuct.ac.za

Abstract: Scrum, a form of agile project management methodology, comes with many benefits derived from the iterative form of software development. Various organisations within South Africa have implemented Scrum within their development teams and are fast receiving positive benefits from it. While Scrum, in many aspects of the methodology, is highly effective and successful within different organisations, there are always going to be pitfalls and negative attributes associated with the adoption and use of a new methodology. The purpose of this research is to determine the factors leading to interruptions in the middle of a sprint while using Scrum. Case studies were conducted which included 3 companies in Johannesburg and Cape Town that have implemented Scrum. In particular, data was collected through the following methods: 12 face-to-face, one-on-one interviews with participating Scrum team members; and 1 group discussion with 8 participating Scrum team members. After analysis, five theoretical statements were formulated pertaining to: poorly understood and defined objectives from clients, management's lack of understanding of Scrum processes, high workload, ad-hoc requests mid-sprint, and low interdepartmental communication. Results from this study reveal the need to conduct possible future research on: ways to prevent these forms of sprint interruption from having negative effects on the Scrum team and the project; and the effect that these interruptions have on the relevant stakeholders involved. The results of the study thus provide managers with the opportunity to take a deeper look into the sources of their Scrum problems and provide them with an understanding as to how they may prevent these interruptions from causing long term, negative effects on the project and the team.

Keywords: Scrum, Sprint, Sources of Interruptions, agile software development, Scrum Teams

1 Introduction

The high rate of software projects failure is one of the major issues faced by the software industry nowadays. Software projects typically overrun because of unrealistic deadlines (West, 2010), which often lead to missed release dates and business opportunities (Ara & Al-Mudimigh, 2011; Cervone, 2011; Keil, Rai, Mann, & Zhang, 2003). Unclear project objectives at the beginning of the project yield scope creep and lack of project focus later during the project lifecycle (Wysocki, 2009). The lack of planning, accountability, feasibility and communication also negatively impacts project success (West, 2010). While there are various reasons behind project failures, they can all be linked to a single denominator, i.e. the lack of effective software project management (Wysocki, 2009).

The traditional approach to software project management has, for a long time, been management driven. Traditional approaches to software development involves getting a clear set of requirements from the client, a timeframe and deadline for completing the requirements and how much they are willing to spend (Wysocki, 2009). A popular approach to software development is the Waterfall methodology (Ara & Al-Mudimigh, 2011). As is the case for the Traditional Software Development Approaches (TSDA), Waterfall follows a specific series of sequential steps: planning; design; development; testing and deployment (Cocco, Mannaro, & Marchesi, 2011).

Several disadvantages have been attributed to TSDA, causing it to be perceived as ineffective pertaining to software project management. When using TSDA, project releases often fall short of the desired goal as the software does not meet the requirements of the client. Clients also have minimal interaction with the team and rarely know what they want from the beginning of a project, which may lead to the development project missing the target dates (Bose, et al., 2008).

Agile Software Development (ASD), like TSDA also follows a project plan. However, that project plan is not as detailed at the beginning of the project (Wysocki, 2009). The plan is designed in stages after the completion of each iteration (Wysocki, 2009). The aim of ASD is to complete the project in less time and at a lower cost than TPM (Cervone, 2011; Sutherland & Altman, 2009; Wysocki, 2009).

Scrum, a form of ASD methodology, comes with many benefits derived from iterative software development. Many organisations in South Africa have implemented Scrum within their software development teams and are reaping benefits from it. In Scrum, the software development cycles are organised in sprints (or iterations) of three to four weeks, during which a potentially shippable set of completed requirements is meant to be produced (Deemer, Benefield, Larman, & Vodde, 2012).

At the beginning of a sprint, the development team identifies a subset of the requirements that will be completed during that iteration. This subset of requirements is a more refined version of what is available from the product backlog (i.e. the overall list of product requirements) (Cervone, 2011; Cocco et al, 2011). In addition, requirements are usually incorporated in the sprint backlog if they are of high priority in the product backlog (Schwaber & Beedle, 2002). The sprint backlog items are often represented as User Stories which are the predominant way for Scrum teams to express features. User stories are short, simple descriptions of the desired functionality told from the perspective of the user. For example: 'As a shopper, I can review the items in my shopping cart before checking out so that I can see what I've already selected' (Schwaber & Beedle, 2002). Once incorporated in the sprint backlog, user stories should not drastically change throughout the sprint to avoid negative impacts on the team's productivity rate and morale (Schwaber & Beedle, 2002). Only minor changes are allowed to the user stories provided that the sprint goal is still achieved.

Customers can exert pressure on the project only at the end of the sprint. During the sprint, customers are not allowed to interfere and the development team is allowed enough time, clearance and trust to perform the work. Any issues reported by the customers at the end of a sprint are addressed in the next sprint (Schwaber & Beedle, 2002). At the beginning of a sprint, the objectives or purpose of that iteration are clearly defined. Customers are also allowed to regularly inspect the product while maintaining little variances from the objectives initially stated. Users are the individual human beings making use of the software application.

Changes to requirements in the middle of a sprint are thus not welcomed by the team as a sprint is not supposed to have any external influences while in progress (Cocco et al., 2011; Rising & Janoff, 2000). Such forms of interruptions should therefore be avoided as they leave the team members in a state of restlessness and uncertainty as to what they are actually supposed to accomplish (Wysocki, 2009). However, many sprints fail because of such forms of interruptions induced by a lack of planning, a lack of team cohesion, and a lack of understanding as to what the team members are actually required to accomplish and what they are responsible for (Wysocki, 2009). Therefore, while Scrum is in many aspects highly effective and successful, practitioners can also experience challenges while engaging with it.

The purpose of this study was to determine the different forms of interruptions which could occur in the middle of a sprint while using Scrum and which lead to unexpected changes in requirements. The research question posed for this study was:

- What are the types of sprint interruptions occurring in the middle of a sprint resulting in unexpected requirements changes?

This study followed an interpretive research philosophy along with an exploratory research purpose. Three case studies were conducted in companies located in Johannesburg and Cape Town (South Africa) that have implemented Scrum. In particular, 12 face-to-face, one-on-one interviews with participating Scrum team members; 1 group discussion with 8 participating Scrum team members; and 8 questionnaires were conducted. The qualitative analysis focusing on the sources of interruptions in the middle of a sprint used an inductive coding approach. Five theoretical statements were formulated pertaining to: *poorly understood and defined objectives from clients, management's lack of understanding of Scrum processes, high workload, ad-hoc requests mid-sprint,* and *low interdepartmental communication*The study also proposes some ways of mitigating these sprint interruptions.

This study is important as it provides Scrum practitioners with useful information on how to identify the sources of interruption during their sprint and proposes ways of managing these interruptions to leverage their chances of achieving software project success. This study also opens up various possible research avenues including research on ways to prevent the sources from negatively affecting the Scrum team, the project, and company as a whole; and the effect that these interruptions have on the relevant stakeholders involved.

2 Literature Review

As part of the literature review, an overview of Scrum is first presented. This is followed by a description of the various forms of interruptions during Scrum sprints as identified from existing literature.

2.1 The Scrum Methodology

Scrum is an adaptive form of APM methodology used for controlling and managing software development projects in an environment that is constantly and rapidly changing (Appan, 2010) (Cervone, 2011). Scrum uses the iterative approach which requires an extensive amount of communication and cooperation from all stakeholders and project team members (Cervone, 2011; Cocco et al., 2011). Scrum is structured around three major components: roles, artefacts, and processes (Cervone, 2011), an overview of which is provided in the following sub-section.

2.1.1 Scrum Roles

Scrum comprises three key roles: the Scrum Master, the Product Owner, the Scrum Team (Keil, et al., 2003). The Scrum master is responsible for guiding the Scrum team towards the production of a successful solution (Deemer et al., 2012). The Scrum Master is not the project manager and is not the team manager. Instead, the Scrum Master simply protects the team and the project from external influences and interferences that may affect the project and the team in a negative way (Cocco et al, 2011; Deemer et al, 2012). The Scrum Master is responsible for resolving issues within the team and project, getting rid of impediments as fast as possible, and ensuring that the Scrum team is on track to meet the requirements of the client (Deemer et al, 2012).

The Product Owner is responsible for determining the scope of the project and transforming the product requirements into a prioritised list which he/she constantly refines as the project progresses (Cervone, 2011; Deemer, et al., 2012).

The Scrum team is self-organized and is responsible for building and developing the product for each iteration or sprint (Cervone, 2011; Cocco et al, 2011; Wysocki, 2009). The Scrum team builds the product whose requirements are determined by the product owner (Deemer et al, 2012).

2.1.2 Artefacts

The Scrum process consists of three main artefacts: the product backlog, the sprint backlog, the burn-down chart and user stories (Cervone, 2011; Cocco et al, 2011). The product backlog consists of all the project requirements that are organised into a prioritized list determined and managed by the Product Owner (Cervone, 2011; Deemer et al, 2012). Essentially the product backlog is a product road-map and is a single view of everything the team will/could be doing at any point in time, in order of priority (Sutherland, Viktorov, Blount, & Puntikov, 2007).

Once the product backlog is ready, the team will break it into sections of work to complete. These sections of work will make up the sprint backlog. The team generally ensures the sprint backlog matches the priority in the product backlog (Sutherland, et al., 2007). Team members start working on the product backlogs' highest priority item first. The sprint backlog, created by the Scrum team, is a refined version of the product backlog, with specific requirements that is to be developed in the sprint (Cervone, 2011; Cocco et al, 2011).

The Scrum team needs to constantly monitor its progress and update the amount of time remaining for the respective tasks. On a daily basis, the team members update their estimate on the time remaining to complete their tasks. These estimates are plotted on a graph known as the burn-down chart. The burn-down chart will show, day by day, a new estimate of time remaining (in person hours) until the tasks are completed. Ideally it slopes downwards to zero as the project moves towards completion (Deemer et al., 2012).

User stories are stored in the Scrum process as sprint backlog items. They consist of an ID, name, importance, initial estimate, demo procedure and notes (Deemer et al., 2012) . User stories are broken down into smaller tasks which Scrum teams have to complete. The initial estimate of stories are be used in the burn-down chart to produce an estimated timeline for the project. In practice these estimates are not always met and will

reflect deviation on the burn-down chart. Generally stories are ordered by level of importance, but can be re-ordered by the Product Owner if necessary.

2.1.3 Scrum Processes

There are six main processes of Scrum: kick-off, sprint planning meeting, the daily Scrum meeting, the sprint review meeting, and the retrospective meeting (Sutherland & Altman, 2009; Wiener, Vogel, & Amberg, 2010). The kick-off refers to the launch of the project. The sprint planning meeting is held to determine the sprint backlog and the sprint objectives, and is attended by the Scrum team, the Scrum Master and the Product Owner (Cervone, 2011; Cocco et al, 2010; Sutherland, & Altman, 2009). In particular, the refined, high priority, items are selected from the product backlog by the Product Owner and the Scrum team and are put together to create a sprint backlog (Deemer et al, 2012).

The daily Scrum meeting is attended by the Scrum master and the Scrum team and lasts approximately 15 minutes (Cervone, 2011; Cocco et al, 2010; Sutherland & Altman, 2009). The purpose of the daily Scrum meeting is to track the progress of each of the Scrum team members and the Scrum team as a whole. Each Scrum team member is asked about what they have done since the last meeting, what they plan to do before the next meeting and are asked to bring up any issues or impediments that are getting in the way of them successfully completing their tasks (Rising & Janoff, 2000).

The sprint review meeting is held by the Product Owner and attended by the Scrum Master, the Scrum team, and the client. This meeting is held after a sprint and the purpose is to demonstrate the functionalities that have been developed during the sprint (Cervone, 2011; Cocco et al, 2011).

The retrospective is a meeting that is held after the sprint review meeting and is attended by the Scrum Master and the Scrum team (Sutherland & Altman, 2009). The retrospective is aimed at identifying the problems experienced during the recent sprint, evaluating the amount of backlog that was covered in the sprint, determining whether or not the Scrum team is still on track, and developing ways of improving future sprints (Sutherland & Altman, 2009).

Interruptions, leading to changes in requirements, throughout a software development project are inevitable and Scrum has been developed to manage these changes in the most efficient and effective way. Changes to requirements in the middle of a sprint, however, are not welcome by Scrum Masters and Scrum teams (Cocco et al, 2011; Rising, & Janoff, 2000). There are, nevertheless, many external influences and interruptions which lead to changes in the requirements and these external influences occur regularly. Changes in requirements during a sprint can negatively affect the Scrum team and the project as a whole. The following sub-section describes sources of interruptions as identified from literature.

2.2 Sources of Interruptions during a Sprint

Literature, particularly experience reports written by practitioners, has identified various forms of disturbances during a sprint. The disturbances can be categorised as follows: Interruptions from the Client, Interruptions from Management, Process-related Interruptions, Communication-related Interruptions, and Experience-related Interruptions. These are further detailed below.

2.2.1 Interruptions from the Clients

Clients, in this context, refer to the internal or external clients of the Scrum team. In particular, clients are unsure of what they actually want (Wysocki, 2009). Clients rarely know what they need from the beginning of the project and it is therefore not uncommon for the client to try and introduce modifications in the middle of a sprint (Cervone, 2011; Wysocki, 2009). As mentioned by Highsmith (2002, p. 4), “projects may have a relatively clear mission, but the specific requirements can be volatile and evolving as customers and development teams alike, explore the unknown”. One of the main complaints by developers is that the constantly changing requirements are difficult to keep up with and are a waste of time (Rising, & Janoff, 2000).

Also, when the user stories are not properly formulated by the client, are vague, or are poorly understandable, the team may find themselves having to change that user story in the middle of the sprint due the fact that they are not on the correct path (Moe, Dingsoyr, & Dyba, 2009).

2.2.2 Interruptions from Management

One of the proponents of ASD relates to the need for self-organised teams (Cockburn, 2002; Ambler, 2005). Moe et al. (2008) use the label "self-organised" teams as a synonym for "autonomous teams" and "empowered teams". Self-organised teams should be responsible for various aspects of their work including planning, scheduling, assigning tasks to members, and making decisions with economic consequences (Guzzo & Dickson, 2006).

The concept of self-organised teams does not equate to uncontrolled teams (Takeuchi & Nonaka, 1986). Relevant check-points should be put in place to prevent ambiguity, tension and instability in the teams but such control should not be enforced so rigidly in that it impairs creativity (Takeuchi & Nonaka, 1986). Moreover, in order for teams to be truly self-managed, they should be able to influence managerial decisions (Tata & Prasada, 2004).

However, in a survey undertaken by Akif and Majeed (2012), it was found that 44% of Scrum team members experienced interruptions from managers during their Scrum projects. In particular, for projects internal to the organisations, management was found to add ad-hoc requirements to the sprint backlog during the sprint (Akif & Majeed, 2012). Such a practice is not in line with Scrum recommendations pertaining to how requirements should be handled during the sprint (Cocco et al., 2011; Rising & Janoff, 2000). Thus, a lack of proper management in Scrum teams can lead to interruptions in the middle of a sprint due to the fact that the team is not effectively managed and is not working efficiently (Schwaber, 1995).

Product Owners and Scrum Masters were also found to at times disrupt the Scrum team by asking for regular statuses as is the case in traditional Software Development Life Cycle (SDLC). This form of disruption prevents the Scrum team from being self-managed (Blankenship, Bussa, & Millet, 2011). Alternatively, a change in management also leads to disruptions in a Scrum team (Blankenship et al., 2011).

2.2.3 Process-Related Interruptions

Processes, in this context, refer to the processes followed by the Scrum teams and within the company as a whole. A loss of resources, such as staff, could be a source of interruption in the middle of a sprint (Stettina & Heijstek, 2011). Losing Scrum team members in the middle of a sprint and having to replace them, often negatively impacts the sprint as it is difficult to accommodate such changes in the middle of a project (Stettina & Heijstek, 2011).

Time pressure is also a source of sprint interruption because if the sprints are not planned and timed properly, the Scrum team could run into trouble leading to the modification of, and possible scrapping of various user stories in the middle of a sprint (Schwaber, 1995).

2.2.4 Communication-Related Interruptions

Communication can be a problem within Scrum teams and management need to ensure that their Scrum teams have all the relevant and necessary tools they need to communicate effectively (Verner, et al., 2012). The six Scrum processes are all geared towards creating a space that enhances and encourages communication during Scrum. These are mostly aligned to one of the Scrum principles which encourages face-to-face communication (Koch, 2005). However, Scrum teams can at times suffer from a lack of effective communication (Sutherland et al., 2007) leading to interuptions.

Other communication-related interruptions occur because of ineffective cross-cultural communication. Cross-cultural communication is also a source of disturbance in the middle of a sprint as the problems are compounded by language barriers, cross-continent time zones, and different work styles that are adapted (Sutherland et al., 2007). Hence, Scrum teams can run the risk of misunderstanding each other, leading to communication breakdowns (Holmström et al., 2006) and sprint interruptions. A lack of shared meaning between the participants can also lead to communication breakdowns. Through shared meaning, Scrum team members can interpret and make sense of each other's actions. Shared meaning is thus further built through face-to-face interactions, over time (BjØrn and Ngwenyama (2009). Communication breakdowns can cause frustrations among team members (Holmström, Fitzgerald, Ågerfalk, & Conchúir, 2006).

While most studies posit that communication should be properly managed and maintained during Scrum, Akif and Majeed (2012) instead highlighted that the increased frequency of communication required by Scrum can be perceived as a form of work interruption by Scrum teams. This is particularly the case due to the frequent meetings which team members perceive as unnecessary.

3 Methodology

This study was interpretive and qualitative in nature. An exploratory research approach was also used to understand the sources of the disturbances experienced by Scrum teams in the case organsiations. Given that no prior hypotheses or propositions were put forward, the study was inductive in nature.

3.1 The Case Study Approach

Three case studies were conducted at three separate companies involved with Scrum and included a detailed investigation of the Scrum processes and problems within each of these organisations. The case study was conducted in a series of 4 steps as shown in Figure 1. Design refers the structuring of the case study and the development of the interview questions. Design also refers to the organising of interviewees at different companies and the arrangement of the interviews and meetings. Conduct refers to the actual interview and meeting process that took place. Transcribe refers to the transcription of all the interviews into one spread sheet. Analyse refers to the in depth analysis that was conducted on the findings and Report refers to the conclusions that were drawn at the end of the research process.

Figure 1: Case Study Design Flow

These case studies included three companies practicing Scrum in Johannesburg and Cape Town. The sample was made up of 20 participants who will, along with the company names, remain anonymous throughout this thesis. Table 5 shows the breakdown of the participants in terms of their company, A, B, or C, their participant number, and their position in the Scrum team. A1 and A2 refer to separate branches within a company.

Participant	Position	Company
1	Product Owner	A1
2	Scrum Master	A1
3	Developer	A1
4	Product Owner	A2
5	Developer	A2
6	Developer	A2
7	Developer	A2
8	Developer	A2
9	Developer	A2
10	Senior Developer	A2
11	Scrum Master	B
12	Developer	B
13	Developer	B
14	Enterprise Application Architect	B
15	Developer	B
16	Test Analyst	B
17	Developer	B
18	Business Analyst	B
19	Product Owner, Business Analyst	C
20	Product Owner, Business Analyst	C

Table 1: Case Study Respondents

3.2 Data Collection

Conducting twenty interviews and one group discussion allowed for a wide range of information, from very different perspectives, to be brought up. Participants included people from each of the different roles within a

Scrum group and this too added to the credibility of the research. All the interviews were recorded and correctly transcribed ensuring that anyone who has access to it can fully understand it. Together, these aspects of the analysis ensure that it is credible, transferable, dependable and confirmable (Thomas, 2006).

The interviews at company A and C were face to face, semi-structured interviews and were conducted on a one on one basis. Each of these interviews lasted approximately twenty minutes. The interviews at company B took the form of a face to face, semi-structured group discussion as well as a questionnaire that was filled out by each of the participants. The Scrum team at company B had limited time available and a group discussion was the most feasible option. The group discussion lasted approximately an hour and the participants were given a few minutes at the end in order to fill out the questionnaires.

3.3 Data Analysis

The data was analysed using the general inductive approach to derive themes and concepts through interpretations of the raw data (Thomas, 2006). The themes and concepts were derived from a series of interview questions that were designed and focused around the research questions. The purpose of the analysis was to summarise the raw data into concepts and identify how these concepts related to each other. Ultimately, theoretical propositions were derived as will be discussed in Section 4. The inductive process also assisted in linking the findings of this research to the literature that already existed on the subject.

3.4 Achieving Validity during the Case Study

Particular attention needs to be placed on reliability and validity related issues while conducting case studies. Construct validity was ensured by gathering data through multiple sources, including 20 semi-structured interviews from three different companies and 1 group discussion which allowed for a wide range of information from different perspectives. The companies are well known companies in South Africa with high professional reputations. The professional reputations of the companies were represented well by the participants. Two of the companies are based in Cape Town and one in Johannesburg. Having a geographical spread of participants provided the richness to the analysis. Participants also included people from each of the different roles within a Scrum group, further adding to construct validity. While coding the constructs throughout the analysis phase, it was also ensured that the concepts could be clearly differentiated from each other. All the interviews were recorded and correctly transcribed ensuring that anyone who has access to it can fully understand it. Together, these aspects of the analysis ensure that it is credible, transferable, dependable and confirmable (Thomas, 2006).

4 Findings

Following the data analysis, seven themes were identified namely: Poorly defined objectives from clients, Management's lack of understanding of the Scrum process, High Workload, External Work, Ad hoc requests from external users from other departments, Ineffective interdepartmental communication, and Team members' lack of experience. Theoretical propositions have also been devised from these themes and are further elaborated below.

4.1 Poorly Understood and Defined Objectives from Clients

In the context of this research study, *Clients* were internal to the organisation and were responsible for providing the Scrum teams with software development projects for use within the organisation itself. In all three case studies, the clients were internal and only few minor projects work was being completed for external customers.

The study found that the objectives put forth by the clients were at times unclear which lead to impromptu requirements change requests in the middle of the sprint once the clients obtained a more precise idea of what they wanted for the project. This is illustrated in the following quotes:

> *"As people get closer to seeing what the final or end product is going to look like, then that's when they start thinking that they actually want it another way"*

> *"… has a lot of problems with people changing what they want in the middle of a sprint"*

The research thus showed that story changes in the middle of a sprint due to the fact that clients change their minds, and do not know what they want from the beginning, have a negative impact on the Scrum team

making the client one of the primary sources of disturbances in the middle of a sprint. The following theoretical statement was formulated from this concept:

- **TS1:** When the sprint objectives are poorly understood and defined by the client, the team could face impromptu requirements change requests mid-sprint

4.2 Management's Lack of Understanding of the Scrum Process

The findings showed that in all three cases, management did not fully understand the processes followed by the Scrum team, which lead them to interfere with the Scrum process by imposing a large number of requirements for the sprint. In addition, the large number of requirements was to be completed within the limited timeframe of a sprint. As mentioned by one participant: *"Management do not push out the deadline. We just have to work harder. We do lots of overtime and sit here until the early hours of the morning and weekends just to try get it all done and cover the workload"*. This statement shows the resulting pressure on the team members because under these circumstances.

Another participant, when speaking about the relationship between top management and the Scrum Master and the Product Owner said that: *"It is very important that the Scrum Master and the Product Owner have a strong relationship with management because otherwise, when things don't get done because we are overloaded, we are left standing in front of the boss explaining why nothing got done".* It is thus important for management to be aware of the Scrum processes so that they can understand the consequences of large number of requirement requests pertaining sprint interruptions induced by team burnout. Such a relationship could emerge from constant communication between management and the key Scrum role players namely the Scrum Master and the Product Owner. The following theoretical statement was formulated for this finding:

- **TS2a:** When Management does not fully understand the Scrum Processes, they are likely to impose large numbers of requirements onto the team which then feels highly pressurised, yielding sprint interruptions

The study also identified that management also tended to put forward ad-hoc requests to the Scrum team, resulting in Sprint interruptions, as seen from this extract: *"Agile is supposed to be flexible on both side but a lot of places it's only flexible in one direction, management direction, management just chop and change what they want, when they want to. Our work is then delayed when this happens"*. Hence, the following theoretical statement could also be formulated:

- **TS2b:** When Management puts forward ad-hoc request to the Scrum team during the sprint, it could result in Sprint interruptions, visible through delayed work.

4.3 Heavy Workload

Related to the interferences emerging from management's lack of understanding of Scrum processes is the issue of the Scrum team's workload. In particular, participants from all three cases stated that they felt that their high workload was a strong source of interference during the sprints. In addition, some team members were working on multiple projects simultaneously and were thus involved in multiple sprints concurrently. This led to many unfinished tasks and a chaotic work ethic within the Scrum team as can be seen in this statement: *"We have had a bad time with some team members in concurrent sprints"… "We tried having them work on concurrent sprints but it didn't work".* A high workload could be problematic as it might cause the Scrum team to miss important deadlines. Given this finding, the following theoretical statement was formulated:

- **TS3a:** When the Scrum team members' workload is too heavy because team members are involved in multiple projects, there could be a high number of incomplete tasks at the end of the sprint

The workload issue was also found to be linked to the estimation process. In particular, when tasks' duration were underestimated at the beginning of the sprint, the workload ended up being too high and the Scrum team were at times incapable of completing everything by the end of the sprint. This is illustrated in the following quotes:

> *"Some of the programmers are still adapting to the concept of estimation and the commitment that that estimation needs. Generally they are doing the work but the estimation might be out from time to time"*
>
> *"I estimated 10 hours and it landed up being 40 just because there were so many changes involved"*

> *"Estimations are quite difficult and not very accurate. It's more of a guestimate. I don't know how much time I really spent on something because bugs pop in and I have to drop what I am doing. It's not an exact science and most of us just guess"*

The study found that, to mitigate the underestimation issue, some developers used buffers while estimating to cater for: the learning curve that needs to happen; bugs and other external interferences; as well as to ease their workload. This is illustrated in the following quote: *"Currently get a task, I add extra hours. I put the buffer in just in case. Testing isn't accounted for so it works for that. Need testing plans to be included because it takes a lot of time".* However, the process of adding large buffers (sometimes up to 100%) had a negative impact on the productivity of the entire Scrum team. The following theoretical statement has been formulated, given this finding:

- **TS3b:** When the Scrum team members' workload is too heavy because of inaccurate tasks estimates, the team could be under stress and is unable to perform efficiently

4.4 Ad Hoc Requests: External Work

In the context of this study, External Work refers to specific work that team members engage in outside the scope of their Scrum project. A prevalent form of external work which the Scrum teams engaged in during this study pertained to bug fixing. Participants from all three companies mentioned that bugs from other systems portfolios, which had already been completed, took up a large portion of their time during the sprint. The bug fixing work was often not taken into consideration during the sprint planning meetings, resulting in Sprint interruptions whenever they occurred, as can be seen in the following quotes:

> *"Bugs take priority over everything"*

> *"Some things need to be done 'now now now!' such as bugs and they take priority even though the sprint is supposed to take priority"*

> *"We have got bugs that pop up quite a bit that are not part of the Scrum. We try to balance it out. Bugs and problems on the live side override everything else. You cannot focus on a task if you know that there is a live issue that clients are having".*

Participants of this study found the need to complete external work to be a major source of interference during Scrum sprint as can be seen in the following quote*: "Most of the time in the retrospective, we analyse what caused the developers to not work all hours. Most of the time it's outside influences and emergencies that we don't have control over".* Given the findings from this study, the following theoretical statement was formulated:

- **TS4a:** When the team faces large numbers of bug fixing requests external to the Scrum project, their ability to complete the Scrum project tasks assigned to them for the sprint could be hindered

The issue of ad hoc bug requests was found to be acerbated when these bug requests were wrongly prioritised. In the cases, it was found that priority was given to bugs and other important aspects that did not relate to the Scrum work from the product backlog. In particular, participants mentioned that:

> *"…Some programmers would see one project as more important than the other"*

> *"Priority shifted and other things such as bugs would take priority and it was a matter of not putting in concrete the actual processes we should have been following".*

Given this finding, the following theoretical statement was formulated:

- **TS4b:** When the reported bugs are incorrectly prioritised (bugs of low priority are marked as being of high priority), the team is unable to complete the tasks assigned to them for the sprint

4.5 Ad hoc requests: External users from other departments

As the project was internal to the organisations in all three cases, the Scrum team often faced ad hoc requests from end users belonging to other departments. These requests did not specifically originate from the official client for the project but from end users themselves. Participants have mentioned that:

> *"Every department has urgent things that they need done and you're in the middle of development and everyone has deadlines. You feel like you don't have time to do it all and you can't please everybody. Other work has to be put on hold to tackle these requests".*

> *"Sometimes when you are on one project and you have a bunch of other stuff to do outside of it and some guy is screaming at you because he wants it done. It causes problems"*

These interview statements show that not only does Sprint work have to be put on hold when these requests are put forward but the Scrum team members also feel pressurised and overwhelmed when trying to please the end users from these various departments. The following theoretical statement has been put forward in light of this finding:

- **TS5:** When Scrum teams are faced urgent ad hoc requests from end-users belonging to other departments within the organisation, their work progress within the Sprint could be hindered

4.6 Low Interdepartmental Communication

The study found that ineffective interdepartmental communication was another source of interference in the middle of a sprint. In the cases investigated, problems occurred when Scrum teams required information from other departments to complete their tasks for the sprint but did not get it due to a breakdown in communication. It was found that often, these departments were unaware that the Scrum team required information from them and therefore did not produce the relevant information on time. This is illustrated from the following statements from the participants:

> *"Big problems with interdepartmental communication"*

> *"There are always problems between departments when you think that a particular department isn't getting things done but everyone also has lots of things that they are busy with"*

> *"On a high level, our teams interact. One team will carry on doing work but they don't take into account that there is something that the other team needs from them in order to carry on. It causes delays".*

Given this finding, the following theoretical statement was put forward:

- **TS6:** When interdepartmental communication is low, work is delayed during the sprint

5 Discussion

The empirical analysis has revealed various forms of interruptions that can occur in the middle of a sprint and which result in unexpected requirements changes. Seven theoretical statements have been formulated to summarise these findings. In this section, each statement is discussed and compared to past literature as a means to extend the debate around forms of interruptions during software development. Suggestions on how to mitigate these interruptions are also provided.

5.1 Poorly Understood and Defined Objectives from Clients

The theoretical statement around poorly understood and defined sprint is specified below.

- **TS1:** When the sprint objectives are poorly understood and defined by the client, the team could face impromptu requirements change requests mid-sprint

Scrum is iterative in nature and requires an extensive amount of communication and cooperation from all stakeholders and project team members (Cervone, 2011; Cocco et al, 2011). Even though it is an agile methodology designed to welcome changes in requirements, these changes are expected to occur in a controlled manner, either at the beginning of a sprint or during the sprint review. Ad hoc changes mid-sprint have been seen to have a negative impact on the project and the team (Wysocki, 2009).

The phenomenon of requirements change requests from clients mid-sprint has also been mentioned in past studies (Rising & Janoff, 2000; Wysocki, 2009). It was also reported that when the user stories are not properly understood or are vague, the team may find themselves having to change the user-story in the middle of the sprint due the fact that they are not on the correct path (Moe et al., 2009). This study extends the existing knowledge by proposing that these changes tend to occur when the clients and the Scrum team do not share a common understanding of the sprint objectives.

Poor understanding of requirements has been widely recognised as having a negative impact on software project success in traditional software development environments (i.e. Waterfall approach) (Ovaska, Rossi & Smolander, 2005). In particular, studies on requirements engineering have identified various issues in that

regard, relating to lack of user involvement, changing, incomplete and ambiguous requirements (Jarke, Rolland, Sutcliffe & Dömges, 1999). While agile software development was devised to mitigate these requirements-related issues, this study reveals that the impact of poorly understood requirements, identified through an inability to articulate a clear sprint objective, can also be felt during Scrum projects. In addition, it further leads to an inability to abide to Scrum rules (Wysocki, 2009).

However, the Scrum team itself should be responsible to nurture and promote a shared understanding of requirements. The study revealed that interruptions occurred when requirements were poorly defined by the clients and not well understood by the team. It is the Scrum team's responsibility (particularly the product owner) to ensure that the sprint's objectives are aligned with the client's expectations and that this objective or goal is understood to all. When this is not achieved, it implies that the Scrum team might also share some responsibility in the subsequent interruptions. This implies that for agile software development, in this case Scrum, to be effective in enabling project success, effort must be made on the part of the project team and customer to have a common understanding of the requirements to be completed for the sprint. This can be achieved through careful backlog grooming prior to the sprint planning meeting as well as more frequent and in-depth discussion between the clients and the product owner. This would allow for a more accurate sprint objective or goal to be formulated, which would further drive the development work during that iteration.

5.2 Management's Lack of Understanding of the Scrum Process

The theoretical statements summarising the impact of management's lack of understanding of Scrum processes have are specified below.

- **TS2a:** When Management does not fully understand the Scrum Process, they are likely to impose large numbers of requirements onto the team which then feels highly pressurised, yielding sprint interruptions
- **TS2b:** When Management puts forward ad-hoc request to the Scrum team during the sprint, it could result in Sprint interruptions, visible through delayed work.

Both theoretical statements indicate a negative impact on the team and the work progress when the level and form of engagement from management is not adequate during Scrum projects. This study particularly highlights the importance of having a management team, well versed in the Scrum processes and the consequences on the project and the team when these processes are not properly implemented.

Improper management practices and senior management unawareness of software development best-practices pertaining to requirements elicitation, have also been reported as an issue in plan-based software development projects. For instance, Verner and Cerpa (2005) reported that 50% of the projects which they investigated had unclear requirements at the beginning as senior management were unaware of the recommended processes to be followed. They further recommended that senior management be better educated regarding the importance of good requirements analysis and schedule estimates. They also recommended that management should take into consideration software developers' input pertaining to task estimation (Verner & Cerpa, 2005). The study reveals that such a challenge can also be felt during Scrum projects.

A lack of understanding of the Scrum processes also reduces the autonomy of the Scrum team, in the sense that they are unable to decide on the number of requirements for the sprint. Such decision should be based on the team's estimation and past velocity. It has been recognized in past literature that self-organised teams does not equate to uncontrolled teams (Takeuchi & Nonaka, 1986). In particular, past studies revealed that an excessive reliance on self-managed teams introduces issues in the Scrum team (Blankenship et al., 2011). A balance should be reached between uncontrolled chaos and too much interference from senior management. As put forward by Tata and Prasada (2004), truly self-managed teams should be able to influence managerial decisions when it comes to the project. This finding supports literature which states that interferences in a project often come from higher management and can disrupt a Scrum team (Keil et al, 2003; Stettina, & Heijstek, 2011). Management thus needs to understand their role within a Scrum team and forego the misconception that in agile projects requirements can change anytime and in an uncontrolled manner as this adds pressure on the Scrum team.

5.3 Heavy Workload

The theoretical statements summarising the impact of team members' workload on sprint interruptions is specified below.

- **TS3a:** When the Scrum team members' workload is too heavy because team members are involved in multiple projects, there could be a high number of incomplete tasks at the end of the sprint
- **TS3b:** When the Scrum team members' workload is too heavy because of inaccurate tasks estimates, the team could be under stress and be unable to perform efficiently

It has been reported in literature that team members are often called upon to work on multiple projects and in multiple teams simultaneously (Leroy & Sproull, 2004; González & Mark, 2004). This is particularly the case in large companies where IT teams from each department are required to provide support for multiple projects, at once (Lindvall, Muthig, Dagnino, Wallin, Stupperich, Kiefer, May, & Kahkonen, 2004). Studies have further shown that in doing so, software team members feel an increase in stress level (Leroy & Sproull, 2004). This study extends the body of knowledge by positing that, in a Scrum development environment, not only is the stress level negatively impacted, but the number of tasks to be completed in the sprint is also reduced.

Past studies on Scrum have also identified workload related issues during Sprints. For instance Schwaber (1995) and Stettina and Heijstek (2011) specified that the loss of team resources in the middle of a sprint further introduces workload issues and time pressures (Beck, 1999). This study further proposes that workload issues, which are relevant forms of interruptions during Scrum sprints, occur because of incorrect task estimations and team members working on multiple projects. These negatively impact on the success of the sprint. Fussell, Kiesler, Setlock, Scupelli and Weisband (2004) further specify that when people work on multiple and equally important projects, they are unable to allocate equal attention and effort to the various relevant tasks. This further increases the stress level of the team members (Leroy & Sproull, 2004).

Studies investigating workload and stress related issues have proposed that successful multitasking can be promoted through more informal, face-to-face conversations as these are perceived as less intrusive (Nardi, Whittaker, & Bradner, 2000). This could be applied to a collocated Scrum environment as a means of alleviating the negative impact of workload on the sprint. Through frequent conversations, Scrum team members can remind each other of project tasks to be completed. In projects where the team members are not collocated, Instant Messenger (IM) was proposed as a solution and was shown to reduce stress (Scupelli, Kiesler, Fussell & Chen, 2005). The use of IM could thus be proposed in distributed Scrum environments, to encourage team members to discuss various projects.

5.4 Ad-hoc Requests Mid-Sprint

The theoretical statements summarising the impact of ad-hoc requests on sprint interruptions is specified below.

- **TS4a:** When the team faces large numbers of bug fixing requests external to the Scrum project, their ability to complete the Scrum project tasks assigned to them for the sprint could be hindered
- **TS4b:** When the reported bugs are incorrectly prioritised (bugs of low priority are marked as being of high priority), the team is unable to complete the tasks assigned to them for the sprint
- **TS5:** When Scrum teams are faced urgent ad hoc requests from end-users belonging to other departments within the organisation, their work progress within the Sprint could be hindered

The importance of having a well-groomed and prioritized product backlog while working in the Scrum environment has been highlighted in many studies (e.g. Moore, Reff, Graham & Hackerson, 2007). Moore et al. (2007) further specified that this is a challenging task because of people's tendency to underestimate the time and effort required to develop the stories for each backlog item. However, a product backlog should also include tasks relative to bug fixing requests. These requests should be prioritized and tackled based in their respective priorities in the upcoming sprints. However, this study reveals that this is not always achievable during Scru, leading to sprint interruptions.

Other studies have also identified that inadequate management of maintenance related work has a negative impact on the sprint, particularly on team productivity and morale (Marchenko & Abrahamsson, 2008). Ad-requests, particularly from the part of the customer have a negative impact on code quality and team's rhythm

(Rayhan & Hague, 2008) and are often underestimated by Scrum teams (Sjøberg, et al., 2012). Often, team members are required to urgently complete the ad hoc bug fixing requests in addition to the existing user stories priorities for the sprint, which further pressurize the team (Sjøberg et al., 2012). However, issues around an inability to manage bug fixing tasks during software projects are not only specific to Scrum (Marchenko & Abrahamsson, 2008).

It is therefore important to put measures in place to ensure that ad hoc bug fixing requests are better managed in Scrum teams. This could be achieved by educating the stakeholders about the need to prioritize bugs and feed them back into the product backlog (Rayhan & Hague, 2008), or by having a strong Scrum Master who is able to shield the team from such external interferences, as recommended by the Scrum methodology (Cocco et al, 2011; Deemer et al, 2012).

5.5 Low Inter-Departmental Communication

The theoretical statements summarising the impact of low inter-departmental communication on sprint interruptions is

- **TS6:** When interdepartmental communication is low, work is delayed during the sprint

Communication is important to software development as a whole and issues around interdepartmental communication and its impact on the sprint has been discussed in literature. For example Cloke (2007) specified that work hand-over between functional departments can be difficult within sprint boundaries.

In the literature, there is more of a focus on the tools and mechanisms that need to be in place in order for effective communication to be possible (Sutherland, et al., 2007). This suggestion is supported by Verner et al (2012) who said that necessary tools are needed in order to communicate effectively. Although the findings of this research do support the need for effective communication tools and mechanisms, there was more of a focus on the particular problems caused by a lack of communication and the source of the problems.

5.6 Proposed Mitigating Strategies

Sprint interruptions, emerge from the client's need for a software product which provides more business value and better aligns to their business needs. From the client's perspective, interruptions are not inherently bad and are instead beneficial. From the software team's perspective, Scrum is a popular software development methodology and in spite of the negative effects of sprint interruptions, the outlook is still mostly positive. In particular, participants mentioned that even though they experienced challenges pertaining to sprint interruptions during the projects, the value of Scrum was still recognized. It can thus be posited that sprint interruptions are an ad-hoc approach for clients to derive business value from the software product and should thus be regarded as an indication that software teams need to dedicate more time to better understand software project requirements to deliver that business value.

This section attempts to propose ways of mitigating the sprint interruptions identified during the study. It is anticipated that further studies would then validate the relevance of these suggestions. The sprint interruptions and the corresponding proposed mitigating strategy are detailed in Table 2.

6 Conclusion

The purpose of this research was to investigate the primary sources of interference in the middle of a sprint. After conducting 12 face-to-face, one-on-one interviews with participating Scrum team members; 1 group discussion with 8 participating Scrum team members; and 8 questionnaires. , the data was analysed to reveal themes around sprint interruptions and the factors leading to them. Five theoretical statements were formulated pertaining to: *poorly understood and defined objectives from clients, management's lack of understanding of Scrum processes, high workload, ad-hoc requests mid-sprint,* and *low interdepartmental communication.* Each of these themes was detailed with reference to quotes from the interviews, and was compared to the current literature. In addition, mitigating strategies were proposed for these sprint interruption occurrences. It is recommended that future studies be executed to validate them. The results of the study thus provide managers with the opportunity to take a deeper look into the sources of their Scrum problems and provide them with an understanding as to how they may prevent these interruptions from causing long term, negative effects on the project and the team.

Table 2: Sprint Interruptions and Proposed Mitigating Strategy

Sprint Interruptions	**Proposed Mitigating Strategy**
TS1: When the sprint objectives are poorly understood and defined by the client, the team could face impromptu requirements change requests mid-sprint	Project team and customer(s) should foster a common understanding of the requirements to be completed for the sprint. For example, during Sprint Reviews, there should be more in-depth discussions on the functionalities and business value derived from current software version between the Scrum team, product owner and client. Only then can accurate objectives and goals be formulated for the next sprint. The product owner should also refine his understanding of the user stories by engaging more regularly with the client and more frequent product backlog grooming. Story definitions need to be thoroughly defined, validated and managed by the Product Owner. Poor defininition and validation of user stories can lead to modifications and therefore impact on the project, client and team.
TS2a: When Management does not fully understand the Scrum Process, they are likely to impose large numbers of requirements onto the team which then feels highly pressurised, yielding sprint interruptions *TS2b: When Management puts forward ad-hoc request to the Scrum team during the sprint, it could result in Sprint interruptions, visible through delayed work.*	Upon implementing Scrum within an organisation, management should be made aware of their expected role as recommended by the methodology, especially pertaining to how requirements should be managed. Management should allow the Scrum Master to perform his/her role correctly. The Scrum Master should ensure that team members and Product Owner work together, not against each other and should shield team members from external forces (customers pushing new requests via the Product Owners) If a large modification is required, it is up to the Product Owner(s) to negotiate with customers on what can be swapped out (something of equal value, in terms of work hours required).
TS3a: When the Scrum team members' workload is too heavy because team members are involved in multiple projects, there could be a high number of incomplete tasks at the end of the sprint *TS3b: When the Scrum team members' workload is too heavy because of inaccurate tasks estimates, the team could be under stress and be unable to perform efficiently*	The Scrum Master should foster informal, face-to-face conversations in order to promote successful multi-tasking when the workload is heavy during the sprint In distributed settings, the Scrum team could use IM to ensure that the team members remind each other of tasks to be completed
TS4a: When the team faces large numbers of bug fixing requests external to the Scrum project, their ability to complete the Scrum project tasks assigned to them for the sprint could be hindered *TS4b: When the reported bugs are incorrectly prioritised (bugs of low priority are marked as being of high priority), the team is unable to complete the tasks assigned to them for the sprint* *TS5: When Scrum teams are faced urgent ad hoc requests from end-users belonging to other departments within the organisation, their work progress within the Sprint could be hindered*	The Scrum Master and Product Owner should educate the stakeholders about the need to prioritize bugs and feed them back into the product backlog Have a strong Scrum Master who is able to shield the team from such external interferences, as recommended by the Scrum methodology The Scrum Master should encourage the team to use Use story points, velocity charts, burn-down charts etc. to determine the boundaries with which story modification and ad hoc requests can occur.
TS6: When interdepartmental communication is low, work is delayed during the sprint	Management should ensure that stakeholders from other departments are informed of deadlines and of the impact of their input on the deliverables to be completed

References

Akif, R. & Majeed, H., 2012. Issues and Challenges in Scrum Implementation. *International Journal of Scientific & Engineering Research,* 3(8), pp. 2229-5518.

Ambler, S. 2005. Quality in an agile world. *SQP, 7*(4), 34-39

Appan, R., 2010. Investigating Retrieval-Induced Forgetting During Information Requirements Determination. *Journal of the Association of Information Systems,* 11(5), pp. 250-272.

Ara, A. & Al-Mudimigh, S., 2011. The Role and Impact of Project Management in ERP Project Implementation Life Cycle. *Global Journal of Computer Science & Technology,* 11(5).

Beck, K., 1999. Embracing Change with Extreme Programming. *IEEE Computer,* 32(10), pp. 70-77.

Blankenship, J., Bussa, M. & Millet, S., 2011. Managing Agile Projects with Scrum. *Pro Agile .NET Development with Scrum,* pp. 13-27.

Bose, S., Kurhekar, M. & Ghosal, J., 2008. *Agile Methodology in Requirements Engineering,* SETLabs Briefings Online, pp. 13-21

Cervone, H. F., 2011. Understanding Agile Project Management Methods using Scrum. *OCLC Systems & Services International Digital Library Perspectives,* 27(1), pp. 18-22.

Cloke, G. 2007. Get Your Agile Freak On! Agile Adoption at Yahoo! Music, *AGILE 2007,* pp. 240-248.

Cocco, L., Mannaro, K. & Marchesi, M., 2011. *Simulating Kanban and Scrum vs. Waterfall with System Dynamics.* Madrid, Spain, s.n., pp. 117-131.

Cockburn, H. 2002. *Agile Software Development.* Boston, MA: Addison-Wesley Professional

Deemer, P., Benefield, G., Larman, C. & Vodde, B., 2012. *The Scrum Primer,* s.l.: s.n.

Fussell, S. R., Kiesler, S., Setlock, L. D., Scupelli, P., & Weisband, S. 2004. Effects of instant messaging on the management of multiple projects. Proceedings of CHI'04, pp. 191-198. NY: ACM Press

González, V.M, & Mark, G. 2004. Constant, constant multi-tasking craziness: managing multiple working spheres. *Proceedings of CHI'04*, pp. 113- 120. NY: ACM Press

Greene, B. 2004. Agile methods applied to embedded firmware development, *Agile Development Conference 2004*, pp. 71-77

Guzzo, R.A. & Dickson, M.W. 1996. Teams in organizations: Recent research on performance and effectiveness. *Annual Review of Psychology, 47*, pp. 307-338.

Highsmith, J., 2002. What is Agile Software Development?. *The Journal of Defence of Software Engineering,* 15(10), pp. 4-9.

Holmström, H., Fitzgerald, B., Ågerfalk, P. J., & Ó. Conchúir, E. 2006. Agile Practices reduce Distance in Global Software Development. *Information Systems Management, 23*(3), 7-18.

Jarke, M., Rolland, C., Sutcliffe, A., and Dömges, R. 1999. The nature of Requirements Engineering, Aachen, Shaker Verlag GmbH.

Keil, M., Rai, A., Mann, J. E. & Zhang, P., 2003. Why Software Projects Escalate: The Importance of Project Management Constructs. *IEEE Transactions on Engineering Management,* 50(3), pp. 251-262.

Koch, A. 2005. *Agile Software Development.* Boston, Massauchusetts: Artech, House

Leroy, S. and Sproull, L. 2004. When team work means working on multiple teams. Unpublished
manuscript, NYU Stern School of Business

Lindvall, M., Muthig, D., Dagnino, A., Wallin, C., Stupperich, M., Kiefer, D., May, J. & Kahkonen, T. 2004. Agile Software Development in Large Organizations, Computer, 37, pp. 26-34

Kniberg, H., 2007. *Scrum and XP from the Trenches,* s.l.: s.n.

Marchenko, A. and Abrahamsson, P. 2008. Scrum in a Multiproject Environment: An Ethnographically-Inspired Case Study on the Adoption Challenges, *Agile 2008 Conference*, pp. 15-26

Moe, N., Dingsøyr, T. & Dybå, T. 2008. Understanding Self-organizing Teams in Agile Software Development, *19th Australian Conference on Software Engineering, IEEE Explore*

Moe, N. B., Dingsoyr, T. & Dyba, T., 2009. Overcoming Barriers to Self-Management in Software Teams. *IEEE Software,* 26(6), pp. 20-26.

Moore, R., Reff, K., Graham, J. & Hackerson, B. 2007. Scrum at a Fortune 500 Manufacturing Company, *AGILE 2007*, pp. 175-180

Nardi, B.A., Whittaker,S., & Bradner E. 2000. Interaction and outeraction: Instant messaging in action. *Proceedings of CSCW'00*, pp. 79-88. NY: ACM Press.

Ovaska, P., Rossi, M. & Smolander, K. 2005. Filtering, Negotiating and Shifting in the Understanding of Information System Requirements, *Scandinavian Journal of Information Systems, 17(1), pp.* 31–66

Pohl, K. 1994. Three Dimensions of Requirements Engineering: Framework and its application, *Information Systems, 19(3), pp. 243-258*

Rayhan, S. H. & Haque, N. 2008. Incremental adoption of Scrum for successful delivery of an IT project in a remote setup, *in Proc. AGILE 2008 Conference, IEEE Computer Society,* Toronto, Canada, 4-8 August 2008, pp. 351-355

Rising, L. & Janoff, N. S., 2000. The Scrum Software Development Process for Small Teams. *IEEE Software,* 17(1), pp. 26-32.

Schwaber, K., 1995. *Scrum Development Process: Advanced Development Methods.* London, UK, s.n.

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

Scupelli, P., Kiesler, S., Fussell, S., & Chen, C. 2005. Project view IM: a tool for juggling multiple projects and teams, *Proceedings of CHI '05 Extended Abstracts on Human Factors in Computing Systems,* pp. 1773-1776

Sjøberg, D. I., Johnsen, A. & Solberg, J., 2012. Quantifying the Effect of Using Kanban versus Scrum: A Case Study. *IEEE Software,* pp. 47-53.

Stettina, C. J. & Heijstek, W., 2011. *Five Agile Factors: Helping Self-Management to Self-Reflect.* Roskilde, Denmark, s.n., pp. 84-96.

Sutherland, J. & Altman, I., 2009. *Take No Prisoners: How a Venture Capital Group Does Scrum..* s.l., s.n.

Sutherland, J., Viktorov, A., Blount, J. & Puntikov, N., 2007. *Distributed Scrum: Agile Project Management with Outsourced Development Team.* Hawaii, IEEE, pp. 274-274.

Takeuchi, H., & Nonaka, I. 1986. The New New Product Development Game, *Harvard Business Review, vol. 64, pp. 137-146, 1986*

Tata, J. & Prasad, S. 2004. Team Self-management, Organizational Structure, and Judgments of Team Effectiveness, *Journal of Managerial Issues, 16(2), pp. 248-265*

Thomas, D. R., 2006. A General Inductive Approach for Analysing Qualitative Evaluation Data. *American Journal of Evaluation,* 27(2), pp. 237-246.

Verner, J.M., & Cerpa, N. 2005. Australian Software Development: What Software Project Management Practices Lead to Success? *Proceedings of the 2005 Australian Software Engineering Conference (ASWEC'05)*, pp. 70-75

Verner, J. & Cerpa, L. 2012. *Risk Mitigation Advice for Global Software Development from Systematic Literature Reviews,* Staffordshire, UK: School of Computing and Mathematics, Keele University.

West, C. K., 2010. *Four Common Reasons Why Projects Fail,* s.l.: s.n.

Wiener, M., Vogel, B. & Amberg, M., 2010. Information Systems Offshoring - A Literature Review and Analysis. *Communications of the Association for Information Systems,* 27(25), pp. 455-492.

Wysocki, R. K., 2009. Effective Project Management - Traditional, Agile, Extreme. In: s.l.:Wiley Publishing Inc..

Yin, R. K., 2003. *Case Study Research, Design and Methods.* Beverly Hills, CA: Sage Publications.