6-18-2013

Scrum Abandonment in Distributed Teams: A Revelatory Case

Paul Ralph
Lancaster University, paul@paulralph.name

Petr Shportun
Bloomberg LP, pshportun1@bloomberg.net

SCRUM ABANDONMENT IN DISTRIBUTED TEAMS: A REVELATORY CASE

Paul Ralph, Lancaster University, Lancaster, UK, paul@paulralph.name

Petr Shportun, Bloomberg LP, London, UK, pshportun1@bloomberg.net

Abstract

The last decade has witnessed substantial growth in the adoption of both Agile and distributed software development. However, combining Agile practices, which emphasize regular informal communication, with geographically and temporally distributed sites, which hinder regular informal communication, presents numerous challenges. Proponents of Agile, especially the Scrum project management framework, have published several case studies of successful Scrum implementations in distributed environments. However, few empirical studies examine failed or abandoned Scrum implementations. Consequently, this paper presents a revelatory case study of a geographically and temporally distributed software development team that abandoned its attempted transition to Scrum. Two factors associated with the team's decision to abandon Scrum are identified – degradation of Scrum practices due to distribution and the undermining of the ScrumMaster's credibility. Based on this analysis the paper proposes that task/team familiarity, group cohesion and transactive memory may be combined to understand the relationship between geotemporal distribution, process and performance.

Keywords: Information Systems Development, Case Study, Scrum, Agile, Failure.

1 INTRODUCTION

Software development is increasingly distributed and global (Herbsleb et al. 2001b). Distributed work takes more time than collocated work (Herbsleb et al. 2001a) as it requires more people (Herbsleb et al. 2003) and coordination (Brooks 2010). Furthermore, distributed development is associated with lower overall success rates (Ambler 2008; Standish Group 2009). Consequently, how to mitigate the productivity impact of distributed work has become a key concern in methods literature (cf., Sutherland et al. 2008). Meanwhile, the increasingly popular agile project management framework Scrum (Schwaber 2004) is associated with modest gains in developer productivity (Cardozo et al. 2010). Consequently, Sutherland et al. (2008) advanced "distributed Scrum" to overcome declining productivity in distributed projects. However, non-trivial challenges are evident in applying Scrum, a framework based on regular, informal communication, in a distributed environment where physical and temporal distance impedes informal communication.

Much of the work on (distributed) Scrum specifically describes cases of successful implementations of the framework (e.g., Berczuk 2007; Sutherland et al. 2008). Cases often present some challenges faced by the team and describe ways in which they were overcome. For example, Pries-Heje et al. (2011) found that Scrum's usefulness in distributed projects derives from its effective use of boundary objects, boundary spanners, social integration and coordination mechanisms (see below).

However, the Information Systems Development (ISD) literature appears to lack studies of Scrum failure or abandonment, i.e., situations where Scrum practices are dropped after participants explicit concede the Scrum adoption initiative. (One exception, discussed below, is Karekar et al., 2011.) Operationalizing project failure as a abandonment is helpful as the latter is less ambiguous (Ewusi-Mensah 2003). Exploring factors associated with abandoning Scrum adoption may provide beneficial insights into what can go wrong, how to avoid it, and when Scrum adoption is unlikely to succeed. Furthermore, the recent advancement of "distributed Scrum" despite the obvious challenges of adopting Scrum in distributed environments makes research on Scrum adoption in distributed teams especially timely. This leads to our primary research question, as follows.

Research Question: *What factors are associated with abandoning Scrum adoption in distributed teams?*

To explore this question we begin by reviewing the literature on Scrum, distributed Scrum, transitioning to Scrum and the roles of knowledge, power and trust in Agile software development. We then describe our research methodology and summarize our findings. Next we discuss the theoretical and practical implications of our results. The paper concludes with a summary of its contributions and limitations, and suggestions for future research.

2 CONTEMPORARY UNDERSTANDING OF SCRUM

Scrum is a framework for managing software development projects. It can be applied in situations characterized by sparse requirements, improvisation and where problem framing and problem solving are entangled (Ralph 2010, 2011, 2012, 2013a, 2013b). Scrum consists of prescribed roles, rules, artifacts, meetings, and assumptions.

Rather than a conventional project manager, Scrum teams have two management roles – a *ScrumMaster* who manages the process and a *Product Owner* who manages the product. The third role, simply *The Team,* does the work. The Team may include analysts, programmers and quality assurance specialists. The Product owner is a boundary spanner (Levina et al. 2005), i.e., an individual who facilitates knowledge sharing between domains, and the focal point for vertical coordination (Nidumolu 1995).

One important rule in Scrum is to *time-box* (limit the time of) activities. Scrum development is organized into time-boxed iterations, called *sprints*, which deliver functionality increments. To say a sprint is time-boxed means that it ends when the allotted time is up, regardless of whether development goals are achieved. Scrum meetings (below) are also time-boxed.

Scrum uses boundary objects to facilitate coordination and transparency. A boundary object is something that is simultaneously adaptable enough to serve different needs in different circumstances

and stable enough to maintain a consistent identity. Examples include conceptual models and design diagrams. Specifically, Scrum makes extensive uses of structured lists of tasks:

> *The Product Backlog is a prioritized list of everything that might be needed in the product. The Sprint Backlog is a list of tasks to turn the Product Backlog for one Sprint into an increment of potentially shippable product. A burndown is a measure of remaining backlog over time. A Release Burndown measures remaining Product Backlog across the time of a release plan. A Sprint Burndown measures remaining Sprint Backlog items across the time of a Sprint.* (Schwaber and Sutherland. 2010, p. 5)

Each sprint comprises development time punctuated by meetings. During the *planning meeting*, which is held immediately before the each sprint, the product owner selects tasks from the product backlog while team members estimate the effort required for each task and self-determine who will complete each task. Each day of development starts with a time-boxed (15 minute) *daily scrum* meeting in which team members describe what they have done since the previous Scrum, what they are going to do next and any obstacles faced. Daily scrums facilitate development of shared mental models, which are crucial for team coordination (Espinosa et al. 2001). Scrum also prescribes review meetings where progress is reviewed and retrospective meetings where participants reflect on the Scrum process and engage in method engineering (Brinkkemper 1996).

Scrum is very popular. In a a recent survey (VersionOne 2011) 52% of respondents reported using Scrum and 17% reported using Scrum hybrids with no other methodology having more than 3%. Meanwhile, Cardozo (2010) found 28 academic studies of Scrum between 2000 and 2009 with a "reasonable level of reliability" (p. 3). Scrum was initially recommended for collocated teams (Schwaber and Beedle 2001), and later extended to distributed teams (Sutherland et al. 2007).

2.1 Distributed Scrum

As (collocated) Scrum is based on constant feedback and intra- and extra-team communication, adapting Scrum to distributed environments is a nontrivial challenge. For instance, Abbattista et al. (2008) argued that "Agile and distributed development practices are so different that, when blended together, the key characteristics of the former exacerbate the challenges intrinsic to the latter" (p. 47).

Reviewing the literature on distributed Scrum reveals that while collocated Scrum has a de facto standard formulation as set out in the official Scrum Guide (Schwaber and Sutherland 2010), distributed Scrum is more ambiguous. For the purposes of this paper, "distributed Scrum" refers to any adaptation of Scrum for geographically or temporally distributed teams. Recommended adaptations for distributed Scrum include:

- daily Scrum team meetings of all developers from multiple sites; daily meetings of Product Owner team (Sutherland et al. 2007)
- shorter sprints; greater emphasis on unit and automated tests (Berczuk 2007)
- shared product backlog but independent sprint backlogs; regular travel between sites; perpetual tele-conferencing connections (Sutherland et al. 2008)

Moreover, Sutherland et al. (2008) suggests three forms of distributed Scrum. In "Isolated Scrums," teams in separate locations conduct independent Scrum meetings. In "Distributed Scrum of Scrums" multiple teams retain independent Scrum meetings but their respective ScrumMasters (or team leaders or project managers) have regular face-to-face or virtual meetings. In "fully distributed Scrums" team members at different sites participate in virtual Scrum meetings.

However, distributed Scrum teams still face serious communication issues (Sutherland et al. 2009; Abbattista et al. 2008) including basic communication disruptions due to time differences between sites that inhibit work synchronization, daily scrums, sprint planning, and review meetings. For example, Vax and Michlaud (2008) found that running daily meetings for all developers across three sites becomes nearly impossible due to time zone differences. Similarly, Sutherland et al. (2007; 2008) found that timing issues forced distributed teams to reduce from Scrum meeting frequency from daily to two or three per week. This hindered Scrum's primary mechanism for team coordination and sharing, the daily meeting (Hossain et al. 2009), which is likely to undermine trust and effectiveness (Iacono et al. 1997). Finding time for longer meetings including sprint planning and sprint review poses an even greater challenge.

Finding effective tools to facilitate communication is also challenging. In Berczuk's (2007) case, the team failed to identify a proper tool for communication. They tried Skype video calls, which was not perfect due to bandwidth constraints and other, mostly technical issues. Furthermore, team members are supposed to stand during Scrum meetings to discourage storytelling, excessive technical detail, and exceeding the 15-minute time-box (Yip 2011). Although standing during videoconferences is physically possible, without specialized hardware team members tend to sit in front of their computers.

Moreover, distributing development teams across countries with different cultures may introduce additional difficulties including language- or accent-induced miscommunications (Hossain et al. 2009) and awkwardness when mixing cultures with different levels of power distance (Sutherland et al. 2009). These difficulties may undermine trust and developer satisfaction. Completing an initial development phase in a collocated manner before splitting into multiple locations may increase team familiarity and therefore have lasting positive effects on team effectiveness (Berczuk 2007; Espinosa et al. 2007). Similarly, regular visits of senior team members may help (Hossain et al. 2009). Despite the agile principle "Working software over comprehensive documentation" (Beck et al. 2001), several papers argue that distributed teams require more extensive documentation (e.g., Abbattista et al. 2008; Hossain et al. 2009). However, the extensive communication required by large projects is not generally reduced by greater documentation (Curtis et al. 1992).

While our literature review uncovered several case studies of successful implementations of distributed Scrum (e.g., Berczuk 2007; Sutherland et al. 2008; Lee and Yong 2010), no cases of failed Scrum implementation were found. More generally, Ambler (2010) found that projects using Agile processes have higher success rates than projects using traditional or ad hoc processes, while Ambler (2008) found that collocated teams have higher success rates than distributed teams. Furthermore, adapting Scrum for distributed work appears to depend on *how* geographically, temporally and culturally distributed the team is – an issue rarely addressed in existing literature.

2.1 Transitioning to Scrum

Another key theme in the Scrum literature concerns challenges and practices associated with adopting or transitioning to Scrum, especially from traditional development methods. For example, developers may initially perceive Scrum as an attempt at micromanagement (Cohn and Ford 2003) as managers take greater interest in each feature, leading to sometimes inadvertent resistance (Nerur et al. 2005). Team members may exhibit dissatisfaction with changing roles (Sumrell 2007) including additional developer responsibilities and the ScrumMaster / Product Owner split. Suggestions for overcoming these challenges include encouraging constant communication and transition support for both employees and upper-management (Schatz and Abdelshafi 2005; Lee 2008). Like other change projects, Scrum adoption may benefit from a "champion" who drives the initiative. Moreover, successful transition depends on how Scrum practices are implemented, adapted and followed.

Of course, challenges in transitioning to Scrum may differ depending on what the team is transitioning *from*, e.g., traditional lifecycle or ad hoc development (Fitzgerald, 1997). Lee (2008) analyzed a Scrum transition through the lens of Tuckman and Jensen's (1977) model, which describes team formation in terms of Forming, Storming, Norming, Performing stages. During the Forming stage, teams tend to follow a manager's orders (Hersey et al. 1979). However, during the Storming and Norming stages, frequent communication is needed to resolve past issues and friction. In the absence of sufficient communication, the team may be unable to proceed past the Forming stage. In Lee's case, for instance, the team felt it necessary to move to an office space where they could all sit within shouting distance of each other.

2.2 Knowledge, Power and Trust in Software Development

Knowledge management, including team knowledge, domain knowledge and knowledge transfer to newcomers, is one of the main capabilities that should be supported in development environments (Curtis et al. 1988; Chau et al. 2003). Different methodologies prescribe different mechanisms for knowledge sharing and retention including documents (Parnas 2009; Parnas et al. 1995), use cases (Jacobson et al. 1999), requirements (Royce 1970), responsiveness (Beck et al. 2001), a boundary-spanning product owner (Schwaber and Beedle 2001), budget and schedule (cf., Brooks 2010), risk (Boehm 1988) and interpersonal relationships (Beck 2005). While the problems associated with losing

team knowledge through employee attrition, forgetting and miscommunication are obvious, documents may quickly become out of date, inconsistent and generally neglected. Therefore, Agile processes including Scrum tend to advocate tacit knowledge shared through interpersonal interaction and team awareness (Beck 2005; Carstensen and Schmidt 2003). Teams may facilitate informal knowledge sharing by sharing physical and virtual spaces (Carsteen and Schmidt 2003) and through a variety of technologies including comprehensive bug trackers, wikis and forums (Bowen and Maurer 2002).

More generally, domain familiarity is crucial for designing complex systems (Curtis et al. 1988; Espinosa et al. 2007). Scrum practices designed to promote domain familiarity include on-site customer, review meetings where customers are invited, regular feedback and the Product Owner role.

A comprehensive understanding of distributed Scrum adoption also involves power and trust issues. While a complete review of the extensive literature on change management, group development and trust is beyond the scope of this paper, the key point for our purposes is twofold. First, project actors may rely on various bases of power including coercive, connection, reward, legitimate, referent, information and expert power (Hersey, Blanchard, and Natemeyer 1979). The Product Owner and ScrumMaster ideally rely on expert power rather than coercion; therefore, if expert power is undermined, the team may lose faith in its leaders, jeopardizing the Scrum adoption effort. Second, Agile methods rely on intra-team trust rather than authority to facilitate coordination and on trust between the the team and other project stakeholders rather than fixed-price contracts to manage risk. Trust generally facilitates information and knowledge sharing (Kramer and Tyler, 1996), while Komiak and Benbasat (2006) differentiate between cognitive trust, which may be more important for team-stakeholder interactions and emotional trust, which may be more important within teams. As distributed teams build trust more slowly than collocated teams (Wilson et al. 2006), the importance of trust is enhanced.

2 RESEARCH METHOD

Studying failing or abandoned Scrum implementations is challenging as it requires identifying one or more organizations having at least the following characteristics:

1. operates in a distributed environment
2. attempted to transition to distributed Scrum
3. unambiguously failed to transition to distributed Scrum
4. contains informants who are willing to share their experiences of what may be perceived as a costly mistake
5. has attempted the transition recently enough that the key actors are still reachable and remember what happened

Having previously considered the difficulties of contemporary field research on method abandonment and project failure, when we became aware of a research-friendly company in midst of a Scrum adoption initiative that appeared to be failing, we took advantage of the opportunity and quickly employed a single case study design.

Yin (2009) argues that single case studies are appropriate where the case is revelatory. “Revelatory” does **not** mean that the case revealed an astonishing discovery; rather, it refers to one where “the investigator has access to a situation previously inaccessible to scientific observation” (p. 43). More generally, demanding multiple-case designs by default on the basis that they have greater generalizability represents a misapplication of statistical generalizability to non-statistical, non-sampling research (Lee and Baskerville 2003). Practically speaking, given the restrictiveness of the above criteria and the absence of studies of failed Scrum implementation in the literature, we are fortunate to have identified a single appropriate organization.

Our objective was to explore the events leading up to Scrum abandonment and identify factors that may have contributed to it. Therefore, we adopted an interpretive case study approach (Eisenhardt, 1989) within a critical realist ontology. Our approach is interpretive in that we studied the attempted

Scrum adoption through the meanings ascribed by participants in interviews and observations, which are socially constructed (Myers and Avison 2002).

2.1 The Context

RA is a small software development company with offices in San Mateo, California and St. Petersburg, Russia. RA develops and maintains a single product, a data analysis platform for brand audience management. The platform creates models of different audiences for online content and facilitates use of these models for targeted advertising among other purposes. This product is developed by a single team, which is distributed between the two offices. RA was founded in 2009 with funding from institutional investors and attracted several experienced personnel before being acquired by a larger company in December 2011. Our study took place in July 2011. During this time, upper management was concentrated in the San Mateo office while development activities were concentrated in the St. Petersburg office.

2.2 Data Collection

When data collection began, RA was in the process of dissolving its Scrum adoption initiative. We therefore proceeded by examining documents and artifacts from the Scrum initiative period as well as interviewing key informants and briefly observing their current practices. All interviews were recorded an transcribed for analysis. Interviews conducted in Russian were transcribed into English by one of the authors. Observation notes were taken in English. We also photographed the work area including all remaining physical artifacts associated with the attempted Scrum implementation.

Semistructured interviews followed the interview guide in the Appendix; however substantial deviations from the guide were allowed when interviewees raised unexpected but possibly important concerns. Interview questions were selected to understand the team, the evolution of their process from before the Scrum initiative to the time of the study, and the factors leadings to Scrum abandonment. Questions concerned participants' perceptions of what changes occurred and why, who was driving the changes, and the outcomes. The interview guide was subjected to a two-stage validation process including review by an expert in Scrum methodology followed by pilot interviews with three practitioners from another company. Validation resulted in minor changes to question wording and structure.

All interview transcripts, observation notes, document copies and photographs were organized in a case database to facilitate analysis.

2.3 Data Analysis

We applied an iterative open-coding process (Silverman and Marvasti 2008) similar to that used in Grounded Theory research (Glaser and Strauss, 1967) to the transcripts, documents and observation notes. During open coding, similar codes are grouped into an unspecified number of categories, the names of which become the "themes" or key findings. We also used mind-mapping (Tattersall et al. 2007) to consolidate issues and identify dependencies. Mind mapping is especially helpful in understanding connections between themes, codes and evidence from the case as well as relating evidence to existing literature. As mind maps quickly become too large for print, Figure 1 provides an example mind map segment illustrating relationships between risk factors in and recommended practices for distributed Scrum. Finally, we shared our analysis with several participants to gauge how well it corresponds to their perceptions of the events in question, leading to minor revisions.

3 FINDINGS

3.1 Background

Beginning in late autumn 2010, RA attempted to transition from their existing ad hoc development approach to distributed Scrum. The transition began as a management initiative to improve the transparency of the development process. It required various role changes for existing employees (summarized in Table 1). In the remainder of the paper we use individuals' original titles and Scrum role names interchangeably.

Prior to the Scrum adoption initiative, the team consisted of two architects, three Java developers (one of whom was designated a "Senior Developer"), two user interface (UI) developers and three testers. The testers acted independently, rather than integrating with the team as is common in Scrum. The team was managed by the chief scientist who was directly involved in the development process and later assumed the Product Owner role. The chief scientist, architects and one of the UI developers were located in San Mateo, California while the remaining team members were located in St. Petersburg, Russia. Architect Two left the company during the transition to Scrum.

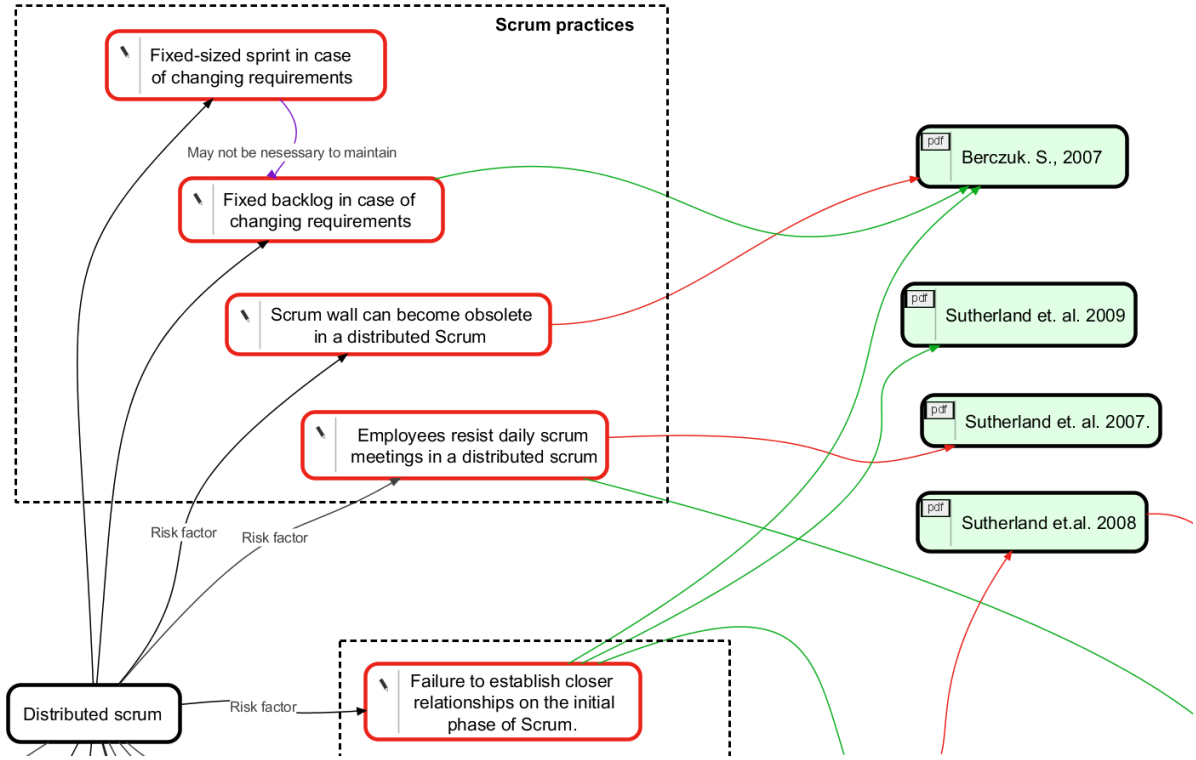

Figure 1. Example Mind Map Segment

Original title	Scrum role	Location	Responsibilities and Activities in Practice
Chief scientist	Product Owner	San Mateo	Managing the development and testing teams including prioritizing tasks and communicating with clients
Architect One	ScrumMaster	San Mateo	Improving the software architecture and introducing Scrum; championing the transition to Scrum
Architect Two	Team Member	San Mateo	Was originally designated an architect but quickly became just a developer albeit senior
Senior Developer	Team Member	St. Petersburg	In practice, the Senior developer retained a de facto leadership role within the team and made many of key technical decisions. Others referred to him as the "Technical Lead"
Developers	Team Members	St. Petersburg	Developing product functionality
UI Developers	Team Members	One in each	Developing product interface
Testers	Testers	St. Petersburg	Product quality assurance

Table 1. Actors and Scrum Roles

Our analysis produced two main themes in participants' perceptions of the factors leading to abandoning the Scrum adoption project: Scrum practice degradation and the undermining of the ScrumMaster. Each of these is described in turn, below. The evidence supporting each theme is summarized in Table 2.

Theme	Example Codes	Example Evidence[1]	PDS[2]
Scrum Practice Degradation	intra-sprint requirements instability	"It is like this: you start developing a feature and you sort out the requirements as you go. And it is all in one sprint, it is not like we fix the requirements and everybody knows exactly what to do at the start." (Developer One) The Product Owner "continued working in the way he did before. So if everything has changed he calls us and says, everything has changed we need to redo the whole thing." (Technical Lead)	No
	Scrum wall redundancy	"Everybody communicates with each other one way or the other and understands the main idea of the work that is being done and there is no need for this sort of reporting." (Developer Two) No one consulted or modified the Scrum Wall during observations. All interviewees indicated that they had stopped using the Scrum wall after a month or so.	Yes
	meeting time conflict	Daily Scrum meetings "were in place. But everyone was unhappy with the time because it was either early in the morning or late in the evening and half the company comes early in the morning and the other half late during the day. So we could not come to a consensus on this question." (Developer One) "it was hard to find time for meetings that suited everybody." (Developer Two)	Yes
Subversion of the ScrumMaster	credibility	The ScrumMaster "started to develop the UI back end himself. And frankly it didn't go well. He was doing it very slowly. There were a lot of bugs, well, there were a lot of problems." (Technical Lead) Developer One made clear that the ScrumMaster, being a newcomer, did not fully understand the project.	No
	infighting	During the interview it was obvious that Technical Lead was charismatic and could manipulate others' opinions. He also spoke quite negatively about work that ScrumMaster did. Other interviewees agreed that Technical Lead was opposing initiatives of the ScrumMaster. The Technical Lead expressed unhappiness that the ScrumMaster had assumed some of his responsibilities, including liaising with the Product Owner.	No
	trust	Several interviewees indicated that the ScrumMaster lost their trust by trying to develop part of functionality himself and doing it badly.	Yes
	relationships	"developers and managers on the USA side did not establish good communication links with the team" (Technical Lead) Developer One indicated that the Technical Lead and ScrumMaster were unable to establish good communication or a good relationship	Yes

Table 2. Challenges in Adopting Distributed Scrum

[1]Quotations translated from Russian with some paraphrasing for clarity
[2]Particular to Distributed Projects (or exacerbated by the distribution of participants)

3.2 Theme One: Scrum Practice Degradation

Berczuk (2007) recommends that during initial Scrum adoption, practices should be implemented by the book, especially when a team has no experience in the process. However, practitioners rarely follow methods precisely or even closely (Mathiassen and Purao 2001). RA initially adopted Scrum roles, sprints, some meetings including daily Scrums and some artifacts including user stories and burndown charts. However, the distributed nature of the team contributed to deterioration of at least three Scrum elements.

First, the daily, time-boxed, stand-up meetings for which Scrum is named are the key facilitator of shared mental models, team cohesion and coordination. RA adopted the "fully distributed Scrum" model (Sutherland et al. 2008) where team members at different locations participate in virtual Scrum meetings. However, the 11-hour time difference between the two sites made it nearly impossible to find meeting times that were convenient for everyone. Consequently, many people regularly missed the meetings, which undermined their usefulness and increased friction within the team. Although Sutherland et al. (2007) report successful daily meetings despite large time zone differences, RA's developers found the meetings disruptive and irritating. This led to communication and knowledge

sharing difficulties (below). Additionally, as meetings were held through Skype, participants tended to sit at their desks, which undermines the key benefits of stand-up meetings, namely, discomfort from standing helps keep meetings short. Consequently, daily meetings tended to exceed their time-box, increasing developers' frustrations.

Second, Scrum calls for a *Scrum Wall* or *sprint burndown chart*, which physically visualizes progress. Tasks are written on sticky notes, which are organized by the developer to which the task is assigned. In contrast, distributed Scrum teams may rely on a digital Scrum wall to coordinate across sites (Sutherland et al. 2007). The physical wall, therefore, simply replicates part of the digital version. In contrast to Berczuk's (2007) case where replicating tasks from an online task tracking tool onto a physical chart was successful, RA's developers perceived the physical version as redundant. One developer explained that the Scrum Wall was only used when the ScrumMaster was present.

Here, the distributed nature of the team necessitated a digital burndown chart, which led to lack of interest in maintaining the physical version. However, the physical wall has benefits that the digital version lacks including increasing work transparency. With a physical chart, anyone with access to the team's premises can see, at a glance, who is working on what and the remaining sprint tasks. "Teams are more likely to see a big, visible chart than they are to look at Sprint burndown chart in Excel or a tool" (Schwaber and Sutherland, 2010).

Third, Scrum calls for development sprints of a fixed and consistent duration (e.g., one month) during which the sprint backlog is fixed; i.e., the Product Owner is not supposed to change requirements during the sprint. However, RA's product owner often altered requirements mid-sprint. In response, the ScrumMaster altered the sprint duration or delayed the next release, citing the intra-sprint changes as justification. The product owner recorded these changes using red sticky notes on the physical burndown chart, but often neglected to update the digital burndown chart. This lead to poor communication between management (in San Mateo) and development (in St. Petersburg). Specifically, the ScrumMaster's justifications for schedule changes referred to the disruptions evident on the physical burndown chart; however, as management could only see the digital burndown chart, delays seemed unwarranted. In this way, the combination of the Product Owner's changes and poor communication caused by physical distribution undermined management's confidence in the ScrumMaster, which relates to the second major theme.

3.3 Theme Two: Subversion of the ScrumMaster

Prior to adopting Scrum, the St. Petersburg group relied heavily on informal communication to facilitate knowledge sharing, with minimal documentation in the form of requirements lists and unstructured notes. The ScrumMaster who was working in San Mateo found that combination of informal communication norms and geotemporal distance inhibited communication and knowledge sharing. Consequently, he transferred to the St. Petersburg office in the third week of the first sprint.

There, the ScrumMaster encountered substantial difficulties in understanding the project, especially its architecture. The developers felt that the ScrumMaster did not sufficiently engage with their informal knowledge sharing to develop a good understanding of the project. However, having previously worked as a software architect, the ScrumMaster made some architectural design decisions. Other team members felt that these decisions were misguided and found them difficult to maintain.

Meanwhile, the formerly entitled "lead developer" was unhappy with his perceived demotion to simply "team member" resulting from adopting Scrum. He consequently invented the title "Technical Lead," opposed the Scrum initiative and antagonized the ScrumMaster. The team and to some extent management perceived the Technical Lead as having greater expertise than the ScrumMaster. The Technical Lead's agonistic approach and the ScrumMaster's poorly perceived architectural revisions combined to undermine the ScrumMaster's credibility with the developers. Meanwhile, the schedule changed discussed in Theme One exacerbated the ScrumMaster's problems by similarly undermining his credibility with upper management.

The ScrumMaster (and newest member of the team) championed Scrum adoption, i.e., he was the intellectual and social force driving the change initiative. When he lost the trust of the development team, he lost the ability to maintain the change (Schatz and Abdelshafi 2005; Komiak and Benbasat 2006). In this way, the Scrum initiative became conflated with the individual advocating it – as the

ScrumMaster's credibility was actively undermined by the Technical Lead, so the whole Scrum implementation was undermined.

4 DISCUSSION AND RECOMMENDATIONS

Karekar et al. (2011) found that a Scrum adoption initiative was abandoned due to "lack of firm leadership commitment to agile, absence of a clearly defined customer ... failure to provide adequate initial or ongoing training and support ... and underestimating the change management requirements" (p. 1). This case differs in that the key factors that drove abandonment appear to be problems implementing Scrum practices due to the team's geotemporal distribution and a conflagration of political infighting, technical errors and resistance to change that undermined the ScrumMaster and project champion's credibility.

Our results also contrast sharply with existing cases of successful distributed Scrum implementation (e.g., Berczuk 2007; Sutherland et al. 2008, 2009; Lee and Yong 2010). More specifically, although Sutherland et al. (2007) and Woodward et al. (2010) argue that teleconferencing can overcome problems associated with daily meetings across time differences, our results suggest that this requires highly motivated participants. Additionally, while some studies (e.g., Sutherland et al. 2008; Hossain et al. 2009) found that good tools could overcome communication issues, this case illustrates how the limitations of basic teleconferencing tools hindered daily meetings and undermined Scrum adoption. Similarly, although Berczuk (2007) found that simply copying information from project management software onto a physical Scrum wall was useful, the RA team denigrated the Scrum Wall as ridiculous. Finally, this case differs from many existing studies of distributed Scrum implementation in that it lacked a charismatic and experienced Scrum guru to champion Scrum adoption.

These results relate to and extend existing theories of familiarity, transactive memory and group cohesion. Familiarity may improve performance in software projects in two different ways – *task familiarity* increases performance "by increasing member ability" while *team familiarity* increases performance "by facilitating recognition and utilization of member expertise" (Littlepage et al. 1997, p. 133). Here, task familiarity indicates participants' domain knowledge and skill level while team familiarity is the extent to which team members are aware of each other's knowledge and expertise. Furthermore, while geographic dispersion and team size negatively impact team performance, groups with greater team familiarity are more resistant to these negative effects (Espinosa et al. 2007).

Task/team familiarity are related to the concepts of memory and meta-memory in the Theory of Transactive Memory, which posits that individuals encode beliefs in memory and beliefs about beliefs in "metamemory" (Wegner 1987). A software development team can therefore be seen as a transactive memory system where individuals (specialists) have not only specialist domain knowledge (memory/ task familiarity) but also knowledge of what other team members know (metamemory/team familiarity). Team performance therefore depends on not only individual skills but also individuals' knowing who to ask about topics outside their skills.

Team familiarity and metamemory also relate to *Group Cohesion.* Group cohesion has long been considered an important if not the most important variable for understanding the formation and performance of small groups (Golembiewski 1962; Lott and Lott 1965). However, ambiguity regarding its precise nature remains (Bollen and Hoyle 1990). For example, group cohesion has been defined as "the tendency for a group to stick together and remain united in the pursuit of its goals and objectives" (Carron et al. 1985, p. 124) while perceived cohesion has been defined as "an individual's sense of belonging to a particular group and his or her feelings of morale associated with membership in groups" (Bollen and Hoyle 1990, p. 482). Here we use team cohesion to indicate the extent to which a team acts together as a single agent toward toward shared goals. Team cohesion implies not only team familiarity but also shared mental models, shared purpose, trust and camaraderie.

Our analysis suggests a more complex relationship between geotemporal distribution, process and performance. Many agile and Scrum practices should strengthen transactive memory and increase team familiarity and cohesion. For example, daily meetings facilitate developing shared mental models and keep team members current on each other's activities (Schwaber 2004). Similarly, the Scrum wall builds trust by increasing process transparency and the Product Owner builds a sense of shared purpose by facilitating experience sharing between team members and project stakeholders.

Peer programming and peer code reviews should strengthen transactive memory (Schmidt et al. 2012) by providing opportunities for knowledge sharing.

However, geotemporal distribution hinders these practices. Stand-up meetings become inconvenient. Scrum walls are virtualized. Stakeholder-team communication is hindered. Programmers can only peer with others in the same physical location. Code reviews become forms instead of conversations. Team members have fewer opportunities to share expertise and build cohesion through socialization.

The RA case particularly illustrates how team cohesion was undermined during the transition to Scrum. As explained above, the technical lead and ScrumMaster did not share a common purpose: the latter sought to transition to Scrum while the former sought to undermine the ScrumMaster and transition. When developers complained that the ScrumMaster did not understand the product, they imply a problem in their shared mental model. When the ScrumMaster made controversial architectural decisions, the team lost trust in him. This lead to a relationship dominated by antagonism rather than camaraderie. In other words, during the transition to Scrum, Team Cohesion collapsed, hindering not only development performance but also performance of the Scrum transition itself.

In summary, focusing on team familiarity or transactive memory downplays emotion; e.g., knowing who can solve your current problem is less helpful if that person refuses to speak to you. Focusing on cohesion, meanwhile, downplays cognition; e.g., feeling comfortable enough with your team to admit a serious error is less helpful if you have no idea which person has the knowledge to fix the error. This motivates further research on distributed team performance, specifically theorizing about team performance in terms of familiarity, transactive memory and cohesion.

Based on the above analysis we can make several practical recommendations. First, organizations seeking to transition to Scrum should pay careful attention to team member's perceptions of role changes. Scrum provides a flat organizational model where all types and levels of developers are simply 'the team'. Employees accustomed to more hierarchical levels may experience indignation at the loss of titles including "chief architect," "senior developer," "head of quality assurance" or 'lead designer". Second, organizations should be aware that employees may conflate the benefits of adopting Scrum with personality and technical competence of the individual championing Scrum adoption. Finally, organizations should be aware that geotemporal distribution is a spectrum where practices that empower a team distributed across two offices in neighboring cities may infuriate a team distributed across 12 time zones; e.g., while teleconferencing may overcome communication barriers, it cannot overcome the irritation of going to work in the middle of the night for a stand-up meeting.

These results and recommendation should be interpreted in light of several limitations. First, our analysis is specific to this particular case and may not generalize to dissimilar situations. Moreover, this is obviously an extreme case with an 11-hour time difference and a disgruntled team member actively subverting the Scrum adoption. However, studying outliers is necessary to fully understand a phenomenon (Van de Ven 2007) and in many ways existing studies (e.g., Sutherland et al. 2007; 2008) are also outliers due to the personal involvement of Scrum co-creator Jeff Sutherland. Second, our data collection took place near the end of the Scrum project and is therefore subject to hindsight bias and other memory effects (Pohl 2004). Third, with the available data we cannot definitively establish causality; therefore, we present ScrumMaster subversion and practice degradation as factors associated with Scrum abandonment rather than causes. Finally and less obviously, in this case the ScrumMaster was also championing Scrum adoption; therefore we cannot disentangle the two roles, i.e., it is unclear whether subversion of the ScrumMaster or subversion of the Scrum champion is the critical factor. Common sense suggests neither benefits the transition.

5 CONCLUSION

This paper presents a revelatory case study of a distributed software team that abandoned a concerted attempted to adopt the popular software project management framework, Scrum. We began with the research question, *what factors are associated with abandoning Scrum adoption in distributed teams?* Consequently, the paper's primary contribution comprises two factors associated with abandoning Scrum adoption in distributed teams – Scrum practice degradation and subversion of the ScrumMaster or Scrum Champion. While the available data and research method cannot establish a definitive causal link between these factors and the team's decision to abandon Scrum, practically speaking they are

obvious warning signs that the transition is challenged. This contribution is novel in that these factors differ from factors identified in previous research on Scrum adoption and abandonment.

Additionally, the paper makes a secondary contribution concerning theoretical lenses applied to understand (software development) team performance. Scrum is explicitly designed to increase team performance and substantial evidence indicates that it practically does so (Cardozo et al. 2010). Several theoretical lenses provide mechanisms, including team familiarity, group cohesion and transactive memory, through which Scrum might increase performance. Our analysis suggests that none of these lenses provides a complete picture and that future research may benefit from considering not only cognitive aspects (familiarity; transactive memory) but also emotional aspects (cohesion) in understanding how Scrum practices affect performance.

By contrasting with existing cases of successful implementations of Scrum in distributed teams, this study highlights the need for more research on not only the adaptations to Scrum practices necessary for distributed work but also the antecedents of successful Scrum transition. Our specific results question the view (cf. Woodward et al. 2010) that the challenges of distributed Scrum generally can be overcome by process modification alone and motivate greater study of the role of team familiarity, group cohesion and transactive memory in Scrum adoption and team performance.

6 APPENDIX: INTERVIEW GUIDE

1. What project are you currently working on? What is it about?
2. What is your role in the project?
3. Would you tell me a little about the development process you're currently using?
4. Do you have any kind of retrospective meetings about the process?
5. Have you made any changes to the process lately?
6. How about since the beginning of the project?
7. How did you come to be using your current process?
8. How has the process changed over the course of the project?

   Probes: Specific example of a change, why it was made, how long it took, whom did it affect?
9. How is the current process working for you? Does it need more changes? If so, what kind?
10. Have there been any political issues surrounding process decisions?
11. How closely do you follow methodology guidance?
12. Have you ever just make things look like they were done according to the process after the fact, to keep management happy?

References

Abbattista, F. et al. (2008). Incorporating social software into distributed agile development environments. 23rd IEEE/ACM International Conference on Automated Software Engineering – Workshops, L'Aquila, Italy, 46-51.

Abrahamsson, P., Salo, O., Ronkainen, J., and Warsta, J. (2002). Agile Software Development Methods: Review and Analysis. Espoo: VTT Publications.

Ambler, S. (2010). 2010 Agile Project Success Rates Survey Results. Ambysoft. Available at: http://www.ambysoft.com/surveys/agileSuccess2010.html [Accessed September 1, 2011].

Ambler, S. (2008). Agile Adoption Rate Survey Results. Ambysoft. Available at: http://www.ambysoft.com/surveys/agileFebruary2008.html [Accessed September 1, 2011].

Beck, K. (2005). Extreme Programming Explained: Embrace Change, 2nd edition. Addison Wesley: Boston, MA, USA.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R.C., Mellor, S., Schwaber, K., Sutherland, J., and Thomas, D. (2001). Manifesto for Agile Software Development. Available at: http://agilemanifesto.org/ [Accessed September 1, 2011].

Berczuk, S. (2007). Back to Basics: The Role of Agile Principles in Success with an Distributed Scrum Team. In AGILE 2007, 382-388.
Boehm, B. (1988). A Spiral Model of Software Development and Enhancement, IEEE Computer 21 (5), May, 61-72.
Blackler, F., (1995). Knowledge, Knowledge Work and Organizations: An Overview and Interpretation. Organization Studies, 16 (6), 1021-1046.
Blaxter, L., Hughes, C. and Tight, M. (2006). How to Research. Open University Press.
Bollen, K.A., and Hoyle, R.H. (1990). Perceived Cohesion: A Conceptual and Empirical Examination, Social Forces 69 (2), 479-504.
Bowen, S. and Maurer, F. (2002). Process support and knowledge management for virtual teams doing agile software development. In Proceedings of the 26th Annual International Computer Software and Applications Conference, 1118-1120.
Brinkkemper, S. (1996). Method Engineering: Engineering of Information Systems Development Methods and Tools, Information and Software Technology 38 (4), 275-280.
Brooks, F.P. (2010). The Design of Design: Essays from a Computer Scientist. Addison-Wesley Professional.
Carron, A.V., Widmeyer, W.N., and Brawley, L.R. (1985). The Development of an Instrument to Assess Cohesion in Sport Teams: The Group Environment Questionnaire, Journal of Sport Psychology (7), 244-266.
Cardozo, E., Neto, J., Barza, A., França, A., and da Silva, F. (2010). Scrum and Productivity in Software Projects: A Systematic Literature Review. In Proceedings of the 14th International Conference on Evaluation and Assessment in Software Engineering, Keele University, UK.
Carstensen, P. and Schmidt, K. (2003). Computer supported cooperative work: New challenges to systems design. In Kenji, I., ed. Handbook of Human Factors/Ergonomics. Tokyo: Asakura Publishing, 619–636.
Chau, T., Maurer, F. and Melnik, G. (2003). Knowledge sharing: agile methods vs. Tayloristic methods. In Proceedings of the Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 302-307.
Cohn, M. and Ford, D. (2003). Introducing an agile process to an organization. Computer, 36 (6), 74-78.
Coplien, J.O., (1994). Borland Software Craftsmanship: A New Look at Process, Quality and Productivity. In Proceedings of the 5th Annual Borland International Conference. Orlando, Florida.
Curtis, B., Kellner, M.I., and Over, J. (1992). Process Modeling, Communications of the ACM 35 (9), 75-90.
Curtis, B., Krasner, H. and Iscoe, N., (1988). A field study of the software design process for large systems. Communications of the ACM, 31 (11), 1268-1287.
Eisenhardt, K.M., (1989) Building Theories From Case Study Research. The Academy of Management Review, 14 (4), 532.
Espinosa, A., Kraut, R., Lerch, J., Slaughter, S., Herbsleb, J., and Mockus, A. (2001). Shared Mental Models and Coordination in Large-Scale, Distributed Software Development. In Proceedings of ICIS 2001, 513-518.
Espinosa, J.A., Sandra, A.S., Robert, E.K., and James, D.H. (2007). Familiarity, Complexity and Team Performance in Geographically Distributed Software Development, Organization Science 18 (4), 613-630.
Ewusi-Mensah, K. (2003). Software Development Failures, Cambridge, MA, USA: MIT Press.
Fitzgerald, B. (1997). The use of systems development methodologies in practice: a field study. Information Systems Journal, 7 (3), 201–212.
Glaser, BG and Strauss, A., (1967). Discovery of Grounded Theory. Strategies for Qualitative Research, Sociology Press.
Golembiewski, R.T. (1962). The Small Group. University of Chicago Press.
Herbsleb, J.D., and Mockus, A. (2003). An Empirical Study of Speed and Communication in Globally Distributed Software Development. IEEE Transactions on Software Engineering 29 (6), 481-494.
Herbsleb, J.D., Mockus, A., Finholt, T.A., and Grinter, R.E. (2001). An Empirical Study of Global Software Development: Distance and Speed. In Proceedings of ICSE 2001, Toronto, Ontario, Canada: IEEE Computer Society, 81-90.
Herbsleb, J.D., and Moitra, D. (2001). Global Software Development, IEEE Software (18:2), 16-20.
Hersey, P., Blanchard, K.H. and Natemeyer, W.E., (1979). Situational Leadership, Perception, and the Impact of Power. Group and Organization Management, 4 (4), 418-428.

Hossain, E., Babar, M.A. and Paik, H.-young (2009). Using Scrum in Global Software Development: A Systematic Literature Review. In Proceedings of the Fourth IEEE International Conference on Global Software Engineering, 175-184.
Iacono, C.S., and Weisband, S. (1997). Developing Trust in Virtual Teams. In Proceedings of HICCS 1997, 412-420.
Jacobson, I., Booch, G., and Rumbaugh, J. (1999). The Unified Software Development Process. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.
Komiak, S.Y.X. and Benbasat, I. (2006). The effects of personalizaion and familiarity on trust and adoption of recommendation agents. MIS Quarterly., 30 (4), 941-960.
Karekar, C., Tarrell, A., and Fruhling, A. (2011) Agile Development at ABC–What Went Wrong?. In Proceedings of AMCIS 2011, paper 283.
Kramer, R.M. and Tyler, T.R., (1996). Trust in organizations: frontiers of theory and research, Sage Publications.
Lave, J. and Wenger, E., (1991). Situated Learning, Cambridge University Press.
Lee, A.S., and Baskerville, R.L. (2003). Generalizing Generalizability in Information Systems Research. Information Systems Research 14 (3), 221-243.
Lee, E.C. (2008). Forming to Performing: Transitioning Large-Scale Project Into Agile. In Proceedings of Agile 2008, 106-111.
Lee, S. and Yong, H.-S. (2010). Distributed agile: project management in a global environment. Empirical Software Engineering, 15 (2), 204-217.
Levina, N., and Vaast, E. (2005). The Emergence of Boundary Spanning Competence in Practice: Implications for Implementation and Use of Information Systems, MIS Quarterly 29 (2), 335-363.
Littlepage, G., Robison, W., and Reddington, K. (1997). Effects of Task Experience and Group Experience on Group Performance, Member Ability, and Recognition of Expertise, Organizational Behavior and Human Decision Processes 69 (2), 133-147.
Lott, A.J., and Lott, B.E. (1965). Group Cohesiveness as Interpersonal Attraction: A Review of Relationships with Antecedents and Consequent Variables, Psychological Bulletin (64), 259-309.
Myers, M.D., and Avison, D.E. (2002). Qualitative Research in Information Systems: A Reader. Sage.
Nerur, S., Mahapatra, R. and Mangalaraj, G. (2005). Challenges of migrating to agile methodologies. Commuications of the ACM, 48 (5), 72-78.
Nidumolu, S. (1995). The Effect of Coordination and Uncertainty on Software Project Performance: Residual Performance Risk as an Intervening Variable, Information Systems Research 6 (3), 191-219.
Parnas, D.L. (2009). Document Based Rational Software Development, Knowledge-Based Systems 22 (3), 132-141.
Parnas, D.L., and Madey, J. (1995). Functional Documents for Computer Systems, Science of Computer Programming 25 (1), 41-61.
Pries-Heje, L., and Pries-Heje, J. (2011). Agile and Distributed Project Management: A Case Study Revealing Why Scrum Is Useful. In Proceedings of ECIS 2011, Helsinki, Finland.
Pohl, R. (ed.) (2004). Cognitive Illusions. East Sussex, UK: Psychology Press.
Ralph, P. (2010). Comparing Two Software Design Process Theories. In R. Winter, J. L. Zhao, & S. Aier (Eds.), Proceedings of DESRIST, St. Gallen, Switzerland: Springer LNCS 6105, 139-153.
Ralph, P. (2011). Introducing an Empirical Model of Design. In Proceedings of The 6th Mediterranean Conference on Information Systems, Limassol, Cyprus.
Ralph, P. (2012). The Illusion of Requirements in Software Development. Requirements Engineering.
Ralph, P. (2013a). The Two Paradigms of Software Design. arXiv:1303.5938 [cs.SE].
Ralph, P. (2013b). The Sensemaking-Coevolution-Implementation Theory of Software Design. arXiv: 1302.4061 [cs.SE].
Rising, L. and Janoff, N.S. (2000). The Scrum software development process for small teams. IEEE Software, 17 (4), 26-32.
Royce, W.W. (1970). Managing the Development of Large Software Systems: Concepts and Techniques, Proceedings of Wescon.
Schatz, B. and Abdelshafi, I. (2005). Primavera gets agile: a successful transition to agile development. IEEE Software, 22 (3), 36-42.
Schmidt, C., Spohrer, K., Kude, T., and Heinzl, A. (2012). The Impact of Peer-Based Software Reviews on Team Performance: The Role of Feedback and Transactive Memory Systems. In Proceedings of ICIS 2012, Orlando, FL, USA, December.
Schwaber, K. (2004). Agile Project Management with Scrum. Microsoft Press.

Schwaber, K., and Sutherland, J. (2010). The Scrum Guide. Scrum.org. Available: http://www.scrum.org/scrumguides/ [Accessed September 1, 2011].
Schwaber, K., and Beedle, M. (2001. Agile Software Development with Scrum. Prentice Hall.
Standish Group (2009). Chaos Summary 2009. Boston, MA, USA. Available at: http://www.standishgroup.com/newsroom/chaos_2009.php [Accessed September 1, 2011].
Silverman, D., and Marvasti, A.B. (2008). Doing Qualitative Research : A Comprehensive Guide. Thousand Oaks, CA, USA: Sage.
Sumrell, M. (2007). From Waterfall to Agile - How does a QA Team Transition? In Proceedings of AGILE 2007, 291-295.
Sutherland, J., Schoonheim, G. and Rijk, M. (2008). Fully Distributed Scrum: Replicating Local Productivity and Quality with Offshore Teams. In Proceedings of HICSS 2008.
Sutherland, J., Schoonheim, G., Rustenburg, E., & Rijk, M. (2008). Fully Distributed Scrum: The Secret Sauce for Hyperproductive Offshored Development Teams. In Proceedings of Agile 2008, 339–344.
Sutherland, J., Viktorov, A., Blount, J., & Puntikov, N. (2007). Distributed Scrum: Agile Project Management with Outsourced Development Teams. In Proceedings of HICSS 2009.
Takeuchi, H. and Nonaka, I., (1986). The new new product development game. Harvard Business Review, 64 (1), 137-146.
Tattersall, C., Watts, A. and Vernon, S. (2007). Mind mapping as a tool in qualitative research. Nursing Times, 103 (26), 32-33.
Tuckman, B.W. and Jensen, M.A.C., (1977). Stages of Small-Group Development Revisited. Group and Organization Management, 2 (4), 419-427.
Vax, M. and Michaud, S. (2008). Distributed Agile: Growing a Practice Together. In Proceedings of Agile, 2008, 310-314.
Wegner, D. M. (1987). Transactive memory: A contemporary analysis of the group mind, In Theories of group behavior, B. Mullen and G. R. Goethals (eds.), New York: Springer, 185–208.
Wilson, J.M., Straus, S.G. and McEvily, B. (2006). All in due time: The development of trust in computer-mediated and face-to-face teams. Organizational Behavior and Human Decision Processes, 99 (1), 16-33.
Woodward, E., Surdek, S. and Ganis, M. (2010). A Practical Guide to Distributed Scrum, Prentice Hall.
Van de Ven, A.H. (2007). Engaged Scholarship: A Guide for Organizational and Social Research. Oxford, UK: Oxford University Press.
VersionOne. (2011). The State of Agile Development. Available at: http://www.versionone.com/state_of_agile_development_survey/11/ [Accessed September 1, 2011].
Yin, R.K. (2009). Case study research: design and methods, Sage Publications.
Yip, J. (2011). It's Not Just Standing Up: Patterns for Daily Standup Meetings. Available at: http://www.martinfowler.com/articles/itsNotJustStandingUp.html [Accessed September 1, 2011].