Coordination in Distributed Agile Software Development: Insights from a COTS-based Case Study

Full Paper

Jim Buchan
School of Engineering, Computer and Mathematical Sciences
Auckland University of Technology
Auckland, New Zealand
Email: jbuchan@aut.ac.nz

A.B. M. Nurul Afser Talukder
School of Engineering, Computer and Mathematical Sciences
Auckland University of Technology
Auckland, New Zealand
Email: a.talukder@aut.ac.nz

Mali Senapathi
School of Engineering, Computer and Mathematical Sciences
Auckland University of Technology
Auckland, New Zealand
Email: mali.senapathi@aut.ac.nz

Abstract

This study investigates the practices of a development team that uses an Agile system of working where some team members and stakeholders were distributed geographically and temporally. The focus of the investigation was to study the dependencies and related coordination activities as the team collaborated on their work, which was the installation and customization of a complex Commercial Off-The-Shelf (COTS) software system in this case. We collected data by interviewing eight key team members and observing three team meetings over a 2-month period. We made detailed field notes and used thematic analysis to identify the key globally distributed dependencies in the development process. We identify and discuss the coordination mechanisms and tools that address these dependencies, along with the main coordination challenges. We conclude by discussing some ideas and lessons learned by the participants which we expect to be useful for other teams in a similar context.

Keywords: Coordination, COTS, customisation, distributed, agile software development, empirical software research, case study.

1 INTRODUCTION

Most contemporary software system development involves teamwork, with individuals in teams and other teams collaborating to achieve the development work. Inevitably, during the development work, there are inter- and intra-team work dependencies. For example, work from one team member (or team) may rely on the work output, knowledge, or a decision of another team or stakeholder. If these dependencies are poorly coordinated, they can result in work delays (e.g. waiting for another's work output, information or decision) or the need for re-work if incorrect assumptions are made. Coordination in Software Development (SD) can be challenging because of such complex dependencies and the potential high impact of poorly coordinated dependencies. Contemporary approaches to address these coordination difficulties often rely on frequent, face-to-face, closed-loop communication and feedback, such as seen in some of the practices in an Agile system of working (Strode and Huff 2015). If a team is globally distributed, it can be expected that coordination is more challenging because some mechanisms for dealing with dependencies in co-located teams are not available with a distributed team (Talukder et al. 2017). For example, the regular and frequent face-to-face interactions foundational to an Agile process, may be impractical for a distributed team, even virtually, because of time differences or technical shortcomings. In-line with other research, it can be expected that Agile practices may need to be adapted or even new practices adopted in such situations of distributed teams.

In our study the work context selected was the installation and customization of a complex Commercial Off-The-Shelf (COTS) software system. The use of COTS solutions to meet organisational information system needs is a growing trend world-wide, with COTS predicted to be offered by 80% legacy and new vendors by 2020 (Gartner 2018). While previous research (e.g. Li et al. 2006; Morisio et al. 2002; Morisio et al. 2000) has focused on understanding COTS implementations in general, empirical studies on how the coordination activities and mechanisms work in practice, as well as the key challenges, are very limited in the distributed agile COTS context. We pose the following research questions for this context:

RQ1: What are the key dependencies in the development process related to working with distributed resources?

RQ2: What are the corresponding coordination activities and supporting mechanisms and tools? RQ3: What are the key coordination challenges?

In this research, we investigated these issues using an in-depth exploratory case study based on field work and involving data collection from both observation of the team at work (field notes and photographs of artefacts), as well as interviews of team members (interview notes). The data collected were analysed using thematic, content, and artefact analysis techniques to gain insights on significant dependencies and how they were coordinated during their development work.

Section 2 provides a brief background to current understanding of coordination management in distributed teams using an Agile system of working. This is followed by a description of the research methodology used in our study in section 3. We describe the organizational, project and team contexts in Section 4, followed by a discussion of the study's main findings in Section 5. Based on our findings, we discuss some learnings, useful to both practitioners and researchers in section 6. We conclude the paper with section 7.

2 COORDINATION IN DISTRIBUTED AGILE SOFTWARE DEVELOPMENT

Research over the past few decades confirm that coordination in collaborative software development is an enduring challenge (e.g. Malone and Crowston 1994; Faraj and Sproull 2000; Kudaravalli et al. 2017). Such studies confirm the high levels of effort needed to manage the interdependencies between different activities as well as the distributed nature of the expert knowledge and skills to get the development work completed. It is common knowledge that the practices, roles and artefacts associated with an Agile way of working are designed to strongly support coordination and communications between stakeholders throughout different aspects of collaborative software development. For example, Scrum practices such as daily stand-up meetings are known to support coordination through continuous communication and feedback (Hossain et al. 2009). Other practices such as maintaining product and sprint backlog, and visualisation using scrum board support in developing an understanding of what is going on and when, who is doing what and resolve coordination issue (Pries-Heje and Pries-Heje 2011).

Our study builds on the conceptualisation of coordination and its effectiveness in co-located Agile teams by Strode (2012) and her subsequent work on a dependency taxonomy (Strode 2016). Although our study context is different, the types are at a level of abstraction that should be useful in many contexts, and this proved to be the case. The dependency taxonomy is summarised in Table 1, where a dependency is "when the progress of one action relies upon the timely output of a previous action or the presence of some specific thing" (Strode 2016, p35).

Knowledge Dependency				**Process Dependency**		**Resource Dependency**	
When a form of information is required				When one task must be completed before another can begin		When an object is required	
Requirement Domain Knowledge or a requirement is not known	**Expertise** Technical or task information is known only by a particular person or group	**Historical** Knowledge about past decisions is needed	**Task Allocation** Who is doing what and when is not known	**Activity** An activity cannot proceed until another activity is complete	**Business Process** An existing BP causes activities to be carried out in a certain order that can affect progress	**Entity** A resource (person, place or thing) is not available	**Technical** A technical aspect of development affects progress. A technical component must interact with another

Table 1 Strode's Dependency Taxonomy (adapted from Strode 2016, p35, Figure.2)

Following the success of Agile software development, and as a natural extension to co-located teams using an Agile approach who have had to adapt to distributed teams, distributed Agile has become more common in globally distributed teams working on large complex product development (e.g. see Paasivaara and Lassenius 2016; Moe et. al. 2014.). Organisations are striving to blend agile software development methods, such as Scrum, with distributed development to realise the benefits of both (Ramesh et al. 2006). It is uncertain if the coordination mechanisms in an Agile way of working can transfer to a distributed team context since some foundational Agile principles, face-to-face communications for example, are impeded in the distributed context. Globally distributed software development adds an additional layer of complexity to activities needed to coordinate dependencies. As discovered by (Bannerman et al. 2011), this could include: increased coordination costs (because of the need for mediating technologies for example); limited informal contact; and misaligned work practices. Also, (Feiner 2016) notes that common communication modes like video conferencing used to coordinate work and share understanding are often less effective than face-to-face coordination because important context in the surrounding environment may be lost and the natural flow of interaction is often impeded. This uncertainty about which Agile practices are effective in a distributed development environment motivates and strengthens the need to continue building an empirical body of evidence in different contexts, which our study contributes to.

Over the last decade a significant body of research on the challenges and practices of distributed development has emerged (e.g. Šāblis and Smite 2016; Sharp et. al. 2012.). There are several research themes related to distributed dependencies and their coordination, including: tools to support coordination of distributed development teams (e.g. Portillo-Rodríguez et. al. 2012); coordination challenges in distributed *agile* development (Sureshchandra and Shrinivasavadhani 2008; Bick et. al. 2017); and large-scale distributed software development (e.g. Espanosa et. al. 2007). Our study extends this understanding of the dependencies and coordination by focusing on a specific contemporary distributed development context: using an Agile development process, and COTS-based product customization work.

3 RESEARCH METHODOLOGY

A case study research approach was selected for this study since it aligns with our aim of acquiring an in-depth understanding of the occurrence of certain phenomena in a real-world context (Yin 2009). An open invitation to participate in the research was presented at a meeting of a professional network of Agile practitioners. The case organization was selected not only because of their willingness and availability, but also because they were using agile practices where some team members were globally distributed and were involved in a complex software project.

Qualitative data was collected using semi-structured interviews and observation of a team at work. Seven interviews were conducted, with participants from the same development squad. All the interviews were conducted by the first two authors at the organisation's headquarters in Auckland, New Zealand. While one researcher conducted the interview, the other took detailed notes. The interviews lasted from 60 to 90 minutes. The interviewees were asked to describe their roles and responsibilities, their experience and perceptions with the process, main dependencies, mechanisms and the challenges experienced in the COTS project. Audio recordings of all interviews were also captured.

In addition, data in the form of field notes collected during observations of team meetings. Three meetings were observed where at least one team member was working from a remote site: two stand-up meetings (around 30 minutes each) and one sprint planning meeting (around 60 minutes). Other data were collected during the site visits to provide a context for this team as well as to collect further information on coordination-related data. This included sketches and photographs of the work environment and shared artefacts on the walls.

Field notes and transcriptions of audio recordings of the interviews were analysed using thematic analysis following the steps in (Fereday and Muir-Cochrane 2006). This involved the researchers grouping data

items with similar meanings together and giving them a relevant label or code. A data item, in this case, was a snippet of either field notes or an audio transcription that indicated a dependency related to the global context. The dependencies were then classified and grouped using the dependency taxonomy from Strode (Strode 2016). The coordination mechanisms for each dependency were inferred inductively from the data, following the approach in (Crowston 1997). Supporting technologies noted for these coordination activities were extracted from data snippets also, and duplications removed. The use of coordination techniques and supporting technologies were also observed during the field work and served to triangulate and deepen the understanding of the data from the interviews. Challenges associated with the dependencies were also identified from the interview data and coded and classified using thematic analysis involving the same open coding technique and then grouping into themes based on patterns.

Coding and classification of the data were done independently by two of the authors and the final codes and classifications agreed on through discussion and negotiation.

4 THE CASE STUDY CONTEXT

4.1 The case organisation and the team

The case organisation is a medium sized organisation in New Zealand which relies on information systems for the provision of the utility services to customers, as well as managing assets and maintenance. The organisation purchased a COTS ERP system to replace their current system and were 12-months into the planned 2-year installation and customization process at the time of this study. The current system is being transitioned to the new cloud-based Software-as-a-Service (SaaS) product suite. The organisation adopted an Agile way of working 9-months prior to the commencement of this study, which took place in August 2018.

There are seven squads involved in the customisation project, one of which is the Operation and Maintenance (O&M) squad, the subject of this study.

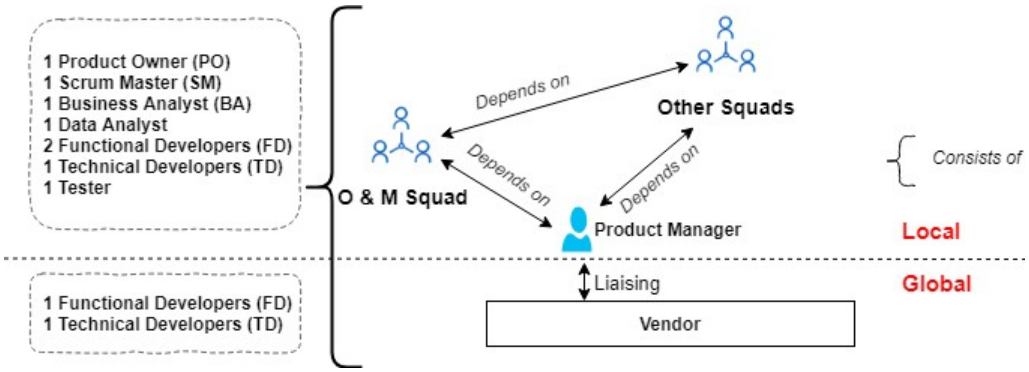

Figure 1 The O&M squad and its coordination context

The O&M squad comprised ten core members (see Figure 1) who are responsible for the configuration and customization of the O&M module in the ERP system. The TD and FD were resources from the vendor with specialised knowledge and worked full-time as part of the O&M squad. All squad members were collocated except the FD and TD who intermittently worked remotely from U.S.A and Singapore respectively for various reasons.

4.2 The COTS Process and Team roles

The O&M squad adhered to most of the core Scrum practices and values, with two-week sprints. An overview of the entire software process for the O&M squad is depicted in Fig 2. The seven squads coordinated their work to deliver a release every twelve weeks (six sprints). The functionality in the release was then deployed and made available to the relevant user group. The work and information dependencies between squads was a challenging coordination issue and significant effort went into managing this, including regular maintenance of a squad dependency board as an information radiator and Scrum-of-scrum meetings. Our study, however, focuses on the interactions and effort to coordinate dependencies between globally distributed entities related to the O&M squad's work.

It was planned that every three sprints newly finished features are released to a group of Beta testers who provide feedback for refinement of the features, data and User Interface (UI) for future sprints in the release.

Requirements discovery was continuously undertaken but was more intensive for the two weeks prior to the start of a new release. Business requirements were identified and clarified from other parts of the

business including SMEs, end-users, vendor representatives and the original Request for Proposal documents. Squad POs coordinated and cooperated to manage these business requirements and order them into release cycles. The O&M squad PO then transformed the business requirements for the next release into user story Epics and grouped them by the personas previously identified. Each persona represented a user whom may be affected by that customisation. The BA then converted the Epics into finer-grained user stories and explicitly identified inter-squad dependencies for each story. These user stories form the Product Backlog for this release and are then prioritized into six sprints by the O&M Squad during Sprint planning.

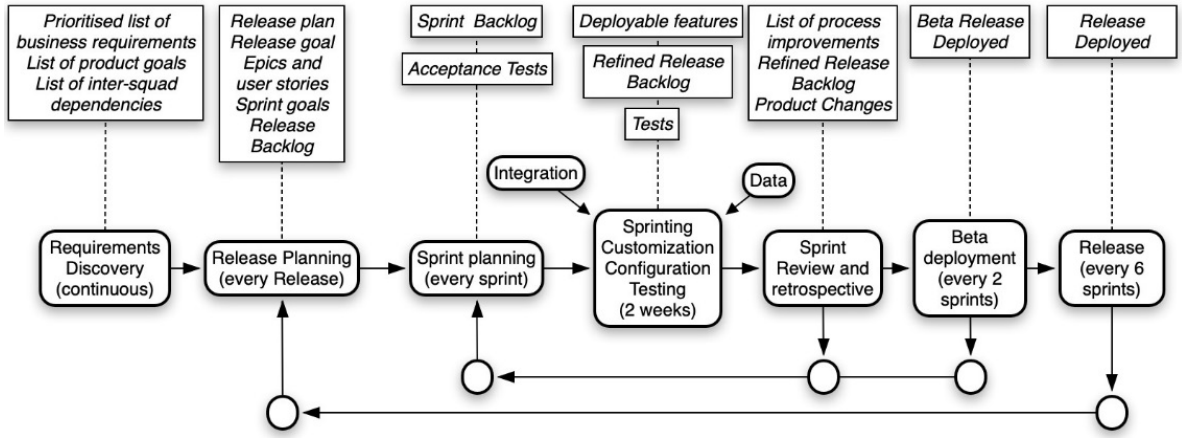

Figure 2 Overview of the COTS customization process

During the sprinting phase in Fig 2 the main activities were coding customizations, testing features, and creating deployment plans. A lot of sprint information is shared through artefacts on the walls around the workplace (e.g. descriptions of personas, release and sprint goals, definitions of "done" and "ready", and project risks). Jira was used as an electronic story board for workflow transparency. Code related to customizations was shared between squad members for development in an online code repository.

There were separate development, testing, user acceptance testing (UAT) and production environments and customizations were moved from one to the next for final deployment into production. While deployment of features was the responsibility of other technical squads, planning and writing instructions for deployment were the O&M squad's responsibility.

At the end of the sprint, a sprint review was conducted often with some stakeholders attending electronically. Generally, a sprint retrospective was held on the same day as the sprint review and was facilitated by the squad PO. The SM acted in this role for three other squads in addition to the O&M squad, although this did not seem to impact on their availability and responsiveness, so coordination with the SM was not an issue. The SM facilitated most squad meetings and managed the squad's capacity planning (input for sprint planning). The BA coordinated with the PO to write user stories and group them by user personas relevant to the O&M team. The BA and also coordinated some inter-team dependencies related to the Squad's current sprint. The BA also managed the Risks and Issues board used to coordinate an understanding of the squad's risks and issues relevant to the current sprint. Subject Matter Experts (SMEs) with specific domain expertise who are external to the squad also participate in Scrum ceremonies such as sprint meetings. The Product Manager (PM) was responsible for most direct communication with the vendor.

The time zone differences between the New Zealand-based squad members and distributed resources are: (1) 6 hrs for the vendor resources, (2) 10 hrs for remote FD (3) 4 hrs for the remote TD.

5 FINDINGS

This section is organized by categories of distributed dependencies: (1) between the local squad and remote vendor and (2) between the local squad members and the remote squad members.

5.1 Squad-vendor dependencies

Table 2 shows that a number of dependencies with the vendor organization were perceived as important or challenging by the interviewees. Interviewees explained that the O&M squad relied on extensive vendor-side support at times. Only the vendor had the expertise and authority to complete certain tasks such as bug-fixing and product enhancement, as well as cloud-related tasks such as module configuration, cloud infrastructure setup, and database connection. The O&M team's work often depended on the completion of this vendor-side work before they could complete their own work. So delays in getting shared

understanding or delivery of the work at the vendor-side could result in delays with the squad's work. Some of these dependencies could be anticipated at the start of a release and early coordination planned for (e.g. cloud infrastructure set up), but often unanticipated needs required ad hoc communications for coordination.

For a COTS system installation, it is common that the base product does not meet all requirements identified by the client, and some of these will be uncovered, at least at a high level, during the COTS research and decision phase (Ahmed et. al. 2016). During the customization phase, decisions about which more detailed customer requirements are not met sufficiently by the base COTS system rely on knowledge about the features available in the base product. When an important gap in the desired functionality of the COTS system is found, there are generally one of three potential outcomes to coordinate: (1) the vendor agrees to extend the functionality of the product for a more general market

(2) the client decides to customize the COTS system to suit their requirement, or (3) the client accepts that the desired requirement is not worth the effort.

For the first situation, the team's work needs to be coordinated with the time for the vendor to make the decision, which can be a lengthy process, and may be a negative answer. The interviewees noted that this can leave parts of the work in a state of uncertainty until the decision is made, making coordination of other work with dependencies on this decision challenging. They stated that while this was a relatively rare event, there was financial pressure to get functional gaps addressed by the vendor, and impact of the delayed decision could be significant for other work. The second situation was described as the most common, where client-specific customisations were made to meet specific desired requirements, at the client's cost. The need to know how to implement this customisation, which often relied on the knowledge and expertise of the vendor about their product and its customisation capabilities.

	Dependency	**Dependency Type**	**Coordination activities**	**Challenges**
1	Cloud-related tasks	Resource-Technical Knowledge-Expertise Process-Activity	Sprint planning Ad hoc communications	Competing priorities Time zone differences Language and cultural differences Process misalignment
2	Product bug fixes	Process-Activity Resource-Entity	Ad hoc communications	Competing priorities Time zone differences Language and cultural differences
3	Product enhancements	Knowledge-Expertise Process-Activity	Ad hoc communications	Competing priorities Time zone difference Process misalignment
4	Understanding and evaluating requirements	Knowledge-Requirement Knowledge-expertise	Sprint planning Daily stand-ups Ad hoc communications	Competing priorities Time zone differences Unclear who has the knowledge needed
5	Understanding how to implement customization	Knowledge-expertise	Sprint planning Daily stand-ups Ad hoc communications	Competing priorities Time zone differences Unclear who has the knowledge needed

Table 2 Findings about Coordination between Distributed Vendor and Co-located Squad

To add to the coordination complexity, the squad-vendor dependencies could require coordination interactions with several vendor teams including the product development team, the support team and the Cloud Operations team.

Most of the identified dependencies (1,3,4,5) and coordination efforts relate to the vendor-skewed distribution of knowledge about the existing COTS product. Hence Table 2 shows most dependencies that are of type "knowledge-expertise". The coordination challenges of this dependency type mainly relate to well-documented coordination-related information exchange (communication) of distributed teams with differences in time zones, language and culture. In addition, since the squad members did not know (or trust, in a team sense) members of the vendor teams, another challenge was the uncertainty in who was the right person to contact for specific work-related information. This aligns with the findings of (Brede and Šmite 2012). Dependencies 1,2,3 have aspects of coordination with work activities that can only be done by the vendor-side resources hence the classification as "process-activity" types. The misalignment between the Agile process of the squad and the more linear process of the vendor resources was a source of coordination challenge. This may exemplify the findings of (Cataldo et. al 2008) regarding the impact of

social and technical incongruence in software teams. This was compounded by the fact that the vendor resources had other clients' work to do and so their priorities did not always coordinate with the squad's needs. The "knowledge-expertise" label for dependency 3 relates to the vendor's authority to decide to develop a missing client requirement as a new product feature, or not. The other coordination challenge mentioned by interviewees relates to the unavailability of the vendor (due to time zone differences) at the time of need for knowledge dependencies. Related to this was the challenge of potential long delays (a day or more) between knowledge seeking, sharing and clarification interaction sequences due to the time zone difference.

Overall, the interviewees indicated that the dependencies with the vendor were challenging to coordinate and had a high impact on the squad's work flow. The coordination mechanisms relied on the use of email, chat-tools and telephone used mainly asynchronously due to the large time-zone and process differences. This introduced an inherent coordination request-response delay or uncertainty and at times, reported interviewees, they felt "helpless to move forward".

5.2 Dependencies between co-located squad-and remote squad member

For periods of time throughout the project one or the other of the two dedicated vendor developers (the FD and TD) worked remotely from the remainder of the squad that was co-located in New Zealand. The coordination during these periods of team distribution was therefore also a focus of our study. Table 3 summarises the significant dependencies identified by both the interviewed co-located team members and the remote team members.

	Dependency	**Dependency Type**	**Coordination activities**	**Supporting Tools**	**Challenges**
1	The development work by the co- located team members relies on the work of the remote worker, and vice versa.	Process-Activity	Daily scrum meetings Ad hoc communications	Code repository. Email, Chat Phone &Video conferencing	Reduced availability and responsiveness due to time-zone differences
2	Testers' work depends on the completion of customization coding by the remote worker	Process-Activity	Sprint planning Daily stand-ups Ad hoc comms	Code repository Jira	Uncertain progress. Testing becomes a bottleneck
3	Remote customization work relies on external access to the shared development environment.	Resource-technical	Planning for external access to resources	None	Firewall and security were a barrier.
4	The development team relies on the expert product knowledge of the TD and FD	Knowledge-Expertise	Sprint planning Daily stand-ups Ad hoc comms	Chat Phone &Video conferencing	Reduced availability and responsiveness
5	Customization design relies on shared understanding of related business goals	Knowledge-historical	Release planning Sprint planning Ad hoc comms	Email Chat	Increased risk of misunderstanding can result in re-work
6	Scrum activities depend on participation of TD and FD	Resource-Entity	Inclusion in squad meetings	Video conf. Shared story board Shared screen	Reduced availability and responsiveness
7	Feedback and review of work	Process-Activity	Code and deliverable reviews	Video conference Shared screen	Reduced availability and responsiveness

Table 3 Findings related to coordination between co-located and remote squad member

The dependencies identified by interviewees related to (1) the work deliverable co-dependencies ("process-activity" type) between the remote worker and the rest of the team (dependencies 1,2,7); and (2) the coordinated participation of each team member ("resource-entity" type) to be involved in the Scrum process activities (dependency 6). Dependencies 4 and 5 are knowledge dependencies with different information sources and sinks. Dependency 3 was highlighted by both remote workers, mainly because remote access to technical resources they depended on (e.g. databases and customized modules) that were hosted at the New Zealand site had been problematic for both of them, with a high impact on their work flow.

Most of the coordinating activities were team activities that were part of the Scrum process. The challenges of geographical separation were addressed by simulating the presence of the remote worker through video conference and electronically shared artefacts that could be manipulated both locally and remotely. This

relies on the availability of the remote worker at the same time as the local team members. Often this meant coordinating a meeting time that was in an acceptable overlapping time-window in the two time zones.

One remote worker changed his working hours (starting work at 7pm) to coincide with the time zone of the local squad. This not only greatly simplified the coordination of squad meetings but also of coordination-related ad hoc communications and work dependencies both ways, (identified as challenging by all interviewees). The main challenge from the squad's perspective related to the reduced responsiveness and availability of the remote worker, particularly for satisfying ad hoc work or information needs.

In this case study the experiences with the two remote workers was quite different. For one remote worker the squad often found it difficult to get his attention when they required some information, either because he was not at work, or because he tended to get focused on his own work. Even when they did get his attention it would often take some time to get the technology set up to have synchronous communication (e.g. web conferencing). This contrasted with the other remote worker who was generally available and attentive to requests from the squad. Furthermore, he had set up a space dedicated to video conferencing so could be talking with the squad within minutes of a request.

6 DISCUSSION AND LESSONS LEARNED

The dependency types in Strode's taxonomy provide useful insights into the significant dependencies specific to the agile distributed process and COTS-based work context, as well as understanding the coordination challenges and suggesting strategies to address them. This is now discussed in relation to dependencies and coordination challenges with the remote COTS vendor firstly, and then with the remote squad members.

6.1 Coordination with a remote COTS vendor

The significant dependency types identified by interviewees for the distributed squad-vendor relationships were mainly Knowledge and Process types, a function of the COTS product knowledge, decisions and development work needed for squad to progress their work, but that were under the control and authority of the vendor. To coordinate Knowledge dependencies we propose that a system is needed that addresses the main challenges identified: (1) surfaces the squad's information needs in sufficient time for communication delays to be low impact on their current and near-term work; (2) provides a mechanism for communication with vendor resources where identification of the right knowledge source and knowledge negotiation interactions are low latency; (3) provides a mechanism for the squad team members to get to know and trust the appropriate vendor-side contacts ; and (4) provides a presence and availability indicator for key people both in the squad and vendor teams.

To coordinate Process-type dependencies, interviewees stated that it would be useful to have transparency in the progress of vendor work (e.g. bug fixing, product enhancement, infrastructure set- up) that impacts the squad's progress. It was "not knowing" that participants said was frustrating for their own planning. Misaligned work processes and priorities, the other challenges of Process dependencies, could be improved by a practice that regularly promotes active understanding of each side's workflow practices and plans, and in particular the consequent sensitivity of work and information timeliness to future progress.

6.2 Coordination with remote Squad members

The dependencies between the squad and the remote worker were mainly of dependency types Process and Resource, reflecting the squad's work dependencies and role dependencies of the Agile way of working, respectively. One coordination challenge in the Resource-entity dependency type related to the expected participation of the remote worker in the activities related to their Scrum-based agile development process (e.g. daily Scrum meetings, Sprint planning, Sprint reviews, Sprint retrospectives). The main mechanism for addressing this dependency coordination was to simulate the presence of the remote worker using video conferencing and screen sharing. Squad members interviewed noted a number important learning from their experiences.

1. The mediating technologies needed to be low friction to use. This took some experimenting with both at the remote worker's site and the squad's work area. Criteria related to this, identified by interviewees and observed by the researchers include:
    - It is the responsibility of a squad member to make sure that the video equipment is available and working and that the remote worker will be available, prior to any planned team meeting.
    - The signup process to join the video conference is straightforward for the remote worker
    - The video screen is large enough and suitably positioned for the entire team to see and hear the remote worker, so the squad has a constant sense of their presence.

- The squad microphone is sensitive enough so that as squad members speak they can be heard by the remote worker. This proved challenging at times because the squad was sometimes spatially spread out in a large room. The squad resorted to passing a microphone around to members who were speaking, which could be a bit disruptive to the normal flow of conversation.
- From the remote worker's perspective, they emphasised that the technology needed to be simple to be set up and use. One remote squad member left the technology set up at all times in a dedicated space to avoid the setup overhead and lower the risk of it not functioning when needed.

2. A highly visible clock set to the time in the remote worker's time zone increased squad empathy for the remote worker's situation during a squad meeting (generally working at an inconvenient or undesirable time from the remote perspective). The clock also helped to coordinate squad member's expectations regarding response times or availability of the remote worker.
3. A number of squad members indicated that it would have been useful to have an indicator of the presence and availability of the remote worker. This would help with managing ad hoc requests and communications and the uncertainty of response times or availability. The remote workers interviewed felt this could be too obtrusive unless they were in control of the presence indicator. These observations align well with the findings of (Steinmacher, et. al. 2013) who investigate the need for awareness support in distributed software development.
4. Since dependencies between squad members were quite dense (knowledge, roles and work), the need for spontaneous, unplanned coordination was quite common, usually short face-to-face interactions. Time zone differences and equipment set-up delays made this type of short spontaneous interaction challenging. This was largely obviated by one remote worker who synchronised his time at work with the squad's and had a video link always on while he was at work, in a dedicated space. While this may not suit all remote worker's situations, it certainly increased the coordination opportunities and reduced delays in workflow.
5. Remote workers reported that they missed the situational awareness of co-location with the rest of the squad. From a coordination perspective, this included:
   - Over-hearing spontaneous interactions that surfaced a coordination need or misunderstanding that turned out to relevant to their own work or they were able to contribute based on their own experience and knowledge.
   - Visibility of the many artefacts in the work environment that acted as information radiators to coordinate knowledge about previous decisions, principles and information that should be kept front-of-mind.
   - These points relate well to the role of situational awareness in coordination (e.g. Endsley 1995) as well as the importance of workplace ambience (e.g. Mishra et. al. 2012).
6. Remote workers reported delays in their work because they depended on remote access to certain databases and product features behind a firewall. They suggested that this be coordinated prior to them traveling to the remote work location.
7. Lack of mutual trust in the distributed team (e.g. Brede and Šmite 2012) was not an issue in this case since the remote worker had also spent considerable time.

It should be noted that some of the knowledge dependencies were anticipated prior to a task starting. For example, often technical and product knowledge were shared during sprint planning sessions, as user stories and associated tasks were clarified. Other knowledge dependencies became important during tasks as understanding and learning deepened. For example, development of acceptance tests uncovered shortcomings in the understanding of user story, or a technical problem triggered the need for a deeper technical understanding of the product from a scarce knowledge resource.

7 CONCLUSION

In this paper we report our interview and field study of a software development team's dependencies and their coordination with remote resources. We answered RQ1 by identifying the key dependencies related to working with the main remote resources, the squad-vendor dependencies and the squad- remote worker dependencies in this case. These were coded and classified using Strode's dependency taxonomy to provide further insights on the dependencies and the nature of the coordination to manage the dependencies. By identifying the coordination activities and any supporting tools related to the dependencies we answered RQ2. This leads to some suggested improvements and lessons learned from the squad's experimentation with different coordination techniques. The interviews and observations also uncovered some coordination challenges where dependencies were not managed well and delays resulted, addressing RQ3.

Classification of the identified dependencies using Strode's dependency types aligned well with the types of coordination mechanisms and challenges described by participants. Investigating the possibility of Strode's dependency types as a planning or diagnostic tool to identify useful candidate coordination mechanisms for a given context could be the subject of future research. We expect these results to be useful to researchers since it raises a number of questions and possibilities, as well as to practitioners in a similar context as suggestions for heightened dependency awareness and improved coordination.

As with any qualitative study there will be researcher biases that have influenced the interpretation, coding and classification of the collected data. We reduced this risk by (1) triangulating some interview findings with the observed behaviour of the squad (2) merging the independent data analyses of multiple researchers. The context of this case is quite specific, and we would not expect it to be generalisable to other contexts. It is hoped that similar coordination studies will be undertaken to add to the body of empirical evidence in similar and other contexts. The perspective of the vendor has largely been ignored in this study and could provide significant insights to coordination. Similarly, the coordination with other squads and the coordination role of boundary spanning project roles and artefacts could be a fruitful extension to this study.

8 REFERENCES

Ahmed, Z., Kumar, U., and Kumar, V. 2016. "Cots Implementation through Process and Control Perspective: A Canadian Government Case," *Journal of Information Technology Case and Application Research* (18:2), pp. 72-92.

Bannerman, P.L., Hossain, E., and Jeffery, R. 2011. "Scrum Practice Mitigation of Global Software Development Coordination Challenges: A Distinctive Advantage?," in: *Proceedings of the Annual Hawaii International Conference on System Sciences*. pp. 5309-5318.

Bick, S., Spohrer, K., Hoda, R., Scheerer, A., and Heinzl, A. 2017. "Coordination Challenges in Large- Scale Software Development: A Case Study of Planning Misalignment in Hybrid Settings," *IEEE Transactions on Software Engineering* (44:10), pp. 932-950.

Brede, N.M., and Šmite, D. 2012. "Understanding a Lack of Trust in Global Software Teams: A Multiple-Case Study," *Journal of Information Systems Education* (23), pp. 243-257.

Cataldo, M., Herbsleb, J.D., and Carley, K.M. 2008. "Socio-Technical Congruence: A Framework for Assessing the Impact of Technical and Work Dependencies on Software Development," in: *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement - ESEM '08*. p. 2.

Crowston, K. 1997. "A Coordination Theory Approach to Organizational Process Design," *Organization Science* (8), pp. 157-175.

Endsley, M.R. 1995. "Toward a Theory of Situation Awareness in Dynamic Systems," Human Factors Journal), pp. 32-64.

Espinosa, J.A., Slaughter, S.A., Kraut, R.E., and Herbsleb, J.D. 2007. "Team Knowledge and Coordination in Geographically Distributed Software Development", *Journal of Management Information Systems*, 24(1), pp. 135-169

Faraj, S., and Sproull, L. 2000. "Coordinating Expertise in Software Development Teams," Management Science (46), pp. 1554-1568.

Feiner, M.E. 2016. "How Scrum Tools May Change Your Agile Software Development Approach," in: *Software Quality. The Future of Systems- and Software Development*.

Fereday, J., and Muir-Cochrane, E. 2006. "Demonstrating Rigor Using Thematic Analysis: A Hybrid Approach of Inductive and Deductive Coding and Theme Development." *International Journal of Qualitative Methods*, pp. 80–92

Gartner. 2018. "Moving to a Software Subscription Model." Retrieved 2nd Jan 2019, 2018, from https://www.gartner.com/smarterwithgartner/moving-to-a-software-subscription-model/

Hossain, E., Babar, M.A., and Verner, J. 2009. "How Can Agile Practices Minimize Global Software Development Co-Ordination Risks?," *Communications in Computer and Information Science* (42), pp. 81-92.

Kudaravalli, S., Faraj, S., and Johnson, S.L. 2017. "A Configural Approach to Coordinating Expertise in Software Development Teams.," *MIS Quarterly* (41), pp. 43-64.

Malone, T.W., and Crowston, K. 1994. "The Interdisciplinary Study of Coordination," *ACM Computing Surveys* (26), pp. 87-119.

Mishra, D., Mishra, A., and Ostrovska, S. 2012. "Impact of Physical Ambiance on Communication, Collaboration and Coordination in Agile Software Development: An Empirical Evaluation," *Information and Software Technology* (54), pp. 1067-1078.

Moe, N.B., Šmite, D., Šāblis, A., Börjesson, A.-L., and Andréasson, P. 2014. "Networking in a Large-Scale Distributed Agile Project," in: *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. New York, NY, USA: ACM, pp. 12:11-- 12:18.

Morisio, M., Seaman, C.B., Basili, V.R., Parra, A.T., Kraft, S.E., and Condon, S.E. 2002. "Cots-Based Software Development: Processes and Open Issues," in: *Journal of Systems and Software*. pp. 189-199.

Morisio, M., Seaman, C.B., Parra, A.T., Basili, V.R., Kraft, S.E., and Condon, S.E. 2000. "Investigating and Improving a Cots-Based Software Development Process," *ICSE*), pp. 32-41.

Paasivaara, M., and Lassenius, C. 2016. "Scaling Scrum in a Large Globally Distributed Organization: A Case Study," *2016 IEEE 11th International Conference on Global Software Engineering (ICGSE)*: IEEE, pp. 74-83.

Portillo-Rodríguez, J., Vizcaíno, A., Piattini, M., and Beecham, S. 2012. "Tools Used in Global Software Engineering: A Systematic Mapping Review," *Information and Software Technology* (54:7), pp. 663-685.

Pries-Heje, L., and Pries-Heje, J. 2011. "Agile & Distributed Project Management : A Case Study Revealing Why Scrum Is Useful," *Ecis*), p. Paper 217.

Ramesh, B., Cao, L., Mohan, K., and Xu, P. 2006. "Can Distributed Software Development Be Agile?," *Communications of the ACM* (49), p. 41.

Šāblis, A., and Smite, D. 2016. "Agile Teams in Large-Scale Distributed Context : Isolated or Connected ?," in: *Proceedings of the Scientific Workshop Proceedings of XP2016*. p. 10.

Sharp, H., Giuffrida, R., and Melnik, G. 2012. "Information Flow within a Dispersed Agile Team : A Distributed Cognition Perspective," in: *13th International Conference on Agile Software Development: XP2012*. pp. 62-76.

Steinmacher, I., Chaves, A.P., and Gerosa, M.A. 2013. "Awareness Support in Distributed Software Development: A Systematic Review and Mapping of the Literature," *Computer Supported Cooperative Work* (22), pp. 113-158.

Strode, D.E. 2016. "A Dependency Taxonomy for Agile Software Development Projects," *Information Systems Frontiers* (18:1), pp. 23-46.

Strode, D.E., and Huff, S.L. 2015. "A Coordination Perspective on Agile Software Development," *Modern Techniques for Successful IT Project Management*), pp. 64-96.

Strode, D.E. 2012. "A Theory of Coordination in Agile Software Development Projects, PhD retrieved from Researcharchive.Vuw.Ac.Nz.

Sureshchandra, K., and Shrinivasavadhani, J. 2008. "Adopting Agile in Distributed Development," *2008 IEEE International Conference on Global Software Engineering*: IEEE, pp. 217-221.

Talukder, A.B.M.N.A., Senapathi, M., and Buchan, J. 2017. "Coordination in Distributed Agile Software Development : A Systematic Review," in: *28th Australasian Conference on Information Systems (ACIS 2017)*. pp. 1-12.

Yin, R.K. 2009. "Case Study Research : Design and Methods," in: *Applied social research methods series;*. pp. xiv, 219 p.

Acknowledgements

The authors acknowledge the practitioners from the case organisation and their willingness to contribute their time and share their experiences in the interviews, observations and informal discussions that are the basis of this paper. We also acknowledge the Agile Auckland Meetup group for their support in allowing us to present our research proposal, which enabled us to find a suitable research partner.

Copyright: © 2019 Buchan, Talukder & Senapathi. This is an open-access article distributed under the terms of the Creative Commons Attribution-NonCommercial 3.0 Australia License, which permits non-commercial use, distribution, and reproduction in any medium, provided the original author and ACIS are credited.