COORDINATION EFFECTIVENESS IN AN AGILE SOFTWARE DEVELOPMENT CONTEXT

Diane E. Strode, School of Information Management, Victoria University of Wellington, Wellington, New Zealand, diane.strode@vuw.ac.nz

Beverley Hope, School of Information Management, Victoria University of Wellington, Wellington, New Zealand, beverley.hope@vuw.ac.nz

Sid L. Huff, School of Information Management, Victoria University of Wellington, Wellington, New Zealand, sid.huff@vuw.ac.nz

Sebastian Link, School of Information Management, Victoria University of Wellington, Wellington, New Zealand, sebastian.link@vuw.ac.nz

Abstract

Effective coordination is an important factor in successful software projects that has not been fully explored in the context of agile software development. Agile software development projects eschew many traditional practices for achieving coordination leaving a gap in our understanding of exactly how such projects are coordinated and if this coordination is effective. However, before investigating coordination behaviours provided in agile software development, and how effective they are, it is necessary to have an understanding of the concept of ‘coordination effectiveness’. This paper defines ‘coordination effectiveness’ based on existing coordination research and case study interview data drawn from experts in the field of agile and non-agile software development. From this, a definition of coordination effectiveness is developed. Such a clearly defined concept should prove useful for future research when assessing coordination behaviours in agile software development projects and for those investigating other types of project and other classes of system development methodology.

Keywords: Agile software development project, coordination strategy, coordination effectiveness.
1 INTRODUCTION

"Agile development methods are examples of apparently major success stories that seem to have run counter to the prevailing wisdom in information systems (IS) and software engineering" (Agerfalk, Fitzgerald, & Slaughter, 2009, p. 317)

For forty years system development methodologies have been designed for different types of organisation, different types of problem, and different technologies (Avison & Fitzgerald, 2006; Olle, Sol, & Tully, 1983). In the 1990s, a new approach to system development, called agile software development, first appeared (Iivari, Hirschheim, & Klein, 2004). This approach is designed to support flexible, rapid, and effective development under conditions of change, uncertainty, and time pressure (Dyba & Dingsoyr, 2008). Such conditions are common in the current fast-paced business environment (Madsen, 2007) where agile software development has proven efficacy (Baskerville, Pries-Heje, & Ramesh, 2007). Agile software development has fundamentally changed the practice of software development (Abrahamsson, Conboy, & Wang, 2009; Agerfalk, et al., 2009; Baskerville, et al., 2007; Rajlich, 2006), and, among practitioners the popularity of this approach has grown rapidly since it was first introduced (Ambler, 2008).

Long before the advent of agile software development, effective coordination was acknowledged as a critical element in organisations generally, and in software development in particular (Curtis, Krasner, & Iscoe, 1988; Kraut & Streeter, 1995; Van de Ven, Delbecq, & Koenig, 1976). Nidumolu (1995) articulated the problem when he asked, “How can software development projects be coordinated more effectively in the presence of uncertainty?” (p. 213) Agile methods are explicitly designed to deal with change and uncertainty, and yet they de-emphasise traditional coordination mechanisms such as forward planning, extensive documentation, specific coordination roles, contracts, and strict adherence to a pre-specified process. Instead, they favour intensive face-to-face communication and other apparently simple practices. Since agile software development is now well accepted, and has contributed to the success of many software projects, clearly the agile approach embodies effective coordination. However, the form and nature of this coordination is not well understood.

There are two important considerations in any study of coordination. The first is to find out what activities and artefacts in a situation support coordinated action, and the second is to identify the features and characteristics of a highly coordinated state – i.e., what such a state “looks like.” Once these two aspects of coordination are clear, it becomes possible to investigate which activities and artefacts are more, or less, effective at bringing about a highly coordinated state.

This paper is based on a subset of findings from a larger research study examining how software projects are coordinated when producing complex software products using a team-based co-located agile development approach and how this approach contributes to effective project coordination. Decomposing this problem into its two parts requires first an identification of activities and artefacts that contribute to coordination, commonly referred to as coordination mechanisms, within agile projects, followed by an investigation of how these mechanisms affect project coordination. Before investigating the second aspect of this problem, however, it is necessary to have a clear definition and solid conceptual understanding of the concept ‘effective project coordination’. Therefore, this paper addresses the question, what is project coordination effectiveness in the context of agile software development?

This paper follows the guidelines of Eisenhardt and Graebner (2007) for theory building from positivist multi-case study research where the aim is to define concepts and their relationships in a form that makes them suitable for testing in future work. In this paper, the focus is restricted to defining the concept of software project coordination effectiveness in the context of agile software development projects. The findings are based upon a literature review of pertinent coordination research along with data from a multi-case study where experienced software developers working on agile and non-agile projects were interviewed to determine their perspectives regarding the attributes of a well-coordinated software development project.

The paper is organised as follows. First, a brief review of the current state of research on agile software development provides a background to the study. Then a discussion of coordination in the
literature in organisation studies, project management studies, and teamwork follows. A discussion of
the universe of interest of the wider study, and how the concept of coordination effectiveness fits
within that universe is presented. This leads to an initial model of project coordination effectiveness.
The case study research method is described next followed by the findings, including a definition of
the concept of project coordination effectiveness. The paper concludes with a discussion on the
findings, the study limitations, and how further research in this area can make use of this concept of
coordination effectiveness.

2 AGILE SOFTWARE DEVELOPMENT

The term ‘agile software development’ is a catch-all term which includes a variety of agile methods,
each of which is founded on a distinct philosophy and a set of practices for conducting software
engineering and information systems development (Abrahamsson, Warsta, Siponen, & Ronkainen,
2003; Hilkka, Tuure, & Matti, 2005). There are at least 13 different agile methods, the most popular
of which are Extreme programming (XP) and Scrum. Others include Dynamic Systems Development
Method (DSDM), Crystal methods, Adaptive System Development, Lean Software Development,
Feature Driven Development, and Evo (Dyba & Dingsoyr, 2008). These methods conform to a
unifying philosophy published in an ‘agile manifesto’ developed by a number of system development
experts who had designed different methodologies based on similar ideals (Beck, et al., 2001). This
philosophy emphasises the importance of individuals and their interactions, teamwork, production
of early working software, collaboration with customers, and responding effectively to change. This is in
preference to the traditional focus on process, tools, documentation, contract negotiation, and
following plans. Each agile method loosely conforms to this philosophy (Conboy, Fitzgerald, &
Golden, 2005) yet each agile method has a distinct focus (e.g. Scrum for project management, XP for
rapidly producing quality software) and a unique set of work practices and iterative micro-phases.

Before 2005, few rigorous studies were available on agile methods, as reported by Dyba and Dingsoyr
(2008). Since publication of that review, studies using rigorous research methods have explored how
agile software development is used. Detailed ethnographic studies describe how practitioners perform
agile development, particularly XP (MacKenzie & Monk, 2004; Sharp & Robinson, 2004) and case
studies provide insights into agile development in specific contexts such as health care systems
(Fruhling & de Vreede, 2006), software product families (Fitzgerald, Hartnett, & Conboy, 2006), and
in distributed projects (Sarker & Sarker, 2009).

There is a large body of existing theory in organisation studies, management science, and information
systems that could fruitfully be brought to bear so as to better understand the agile approach, and to
explain and predict its effects and efficacy. Cao and Ramesh (2007), Nerur and Balijepally (2007) and
Dingsoyr, Dyba, and Abrahamsson (2008) have all called for this type of work. Some examples
include a study by Conboy (2009) who defined a theory of agility in the context of systems
development by linking agile software development to agility in other fields. In addition, empirically
supported studies linking distributed agile software development and project success concepts have
now appeared (Sarker, Munson, Sarker, & Chakraborty, 2009). One management theory that has been
applied to the distributed agile context is control theory. Maruping, Venkatesh and Agarwal (2009)
use control theory to explain the impact of agile software development on software project quality,
whereas Harris, Collins, and Hevner (2009) extend control theory to explain the effectiveness of
flexible management practices using empirical data from distributed agile projects.

Although research into the agile approach to software development has matured in the last decade and
is increasingly more rigorous and theory-based, many unanswered questions remain and many
concepts relevant to information systems development are not understood in this context
(Abrahamsson, et al., 2009; Agerfalk, et al., 2009). One of these fundamental concepts is
coordination.
3 COORDINATION

When software developers carry out agile software development, they work in organisations, on projects, and within teams. In all three contexts, coordination is critical. Therefore, much existing coordination research in organisation studies, IS projects, and teamwork, may be relevant in understanding this approach. However, before discussing these three contexts, it is necessary to define ‘coordination’ and to describe an existing major ‘theory of coordination.’

No single widely accepted definition of coordination exists. When Malone (1988) first proposed a theory of coordination he defined coordination as “the additional information processing performed when multiple, connected actors pursue goals that a single actor pursuing the same goals would not perform” (1988, p. 5). Later this was refined by Malone and Crowston (1994) when they developed their interdisciplinary theory of coordination which is based on the tenet that, “coordination is the managing of dependencies” (Malone & Crowston, 1994, p. 90). Quite a different tack is taken by knowledge management researchers, who define coordination as a way to manage “problems of sharing, integrating, creating, transforming, and transferring knowledge” (Kotlarsky, van Fenema, & Willcocks, 2008, p. 96). While multiple definitions of coordination are evident in the literature, only a single broad-based theory has been developed and published.

Malone and Crowston (1994) developed a major interdisciplinary theory of coordination, which they termed Coordination Theory. Coordination Theory is based on ideas from organisation theory, economics, management, and computer science. The key idea in Coordination Theory is that coordination is needed to address dependencies, which are the constraints on action in a situation. Coordination is made up of one or more coordination mechanisms; each one addresses one or more dependencies in a situation. While Coordination Theory is very useful for identifying dependencies, categorising those dependencies, and identifying the coordination mechanisms in a situation, it comprises a ‘theory for analysis’ (Gregor, 2006, p. 623), and is not suitable for prediction. In particular, Coordination Theory has no stated propositions or hypothesised relationships and therefore does not meet the strict definition of theory proposed by Bacharach (1989). In their 10-year retrospective of Coordination Theory research, Crowston, Rubleske, and Howison (2006) acknowledge these limitations and stated, “challenges for future research include developing testable hypotheses (e.g. about the generality of coordination mechanisms)” (p. 135). This limitation means that Coordination Theory in its current form cannot be applied to predict outcomes such as coordination effectiveness or project effectiveness.

The bulk of the extant coordination research has been carried out in three specific contexts: organisation studies, IS projects, and teamwork. In the following sections, research in each context is presented in turn, along with a consideration of its relevance to studies of agile software development projects.

Organisation studies is the first context in which coordination is a focus. Coordination is considered a key determinant of organisation structure and various organisation-level coordination modes have been identified (Galbraith, 1977; Malone, 1987; March & Simon, 1958; Mintzberg, 1980; Thompson, 1967; Van de Ven, et al., 1976). Impersonal coordination mode is when “a codified blueprint of action is impersonally specified” (Van de Ven, et al., 1976, p. 323) and is achieved with the use of “pre-established plans, schedules, forecasts, formalized rules, policies and procedures, and standardized information and communication systems” (Van de Ven, et al., 1976, p. 323). Minimal verbal communication is a characteristic of this mode. Coordination by mutual adjustment mode (Thompson, 1967), is either group or personal (Van de Ven, et al., 1976). Scheduled or unscheduled meetings achieve group coordination whereas personal mode is either vertical or horizontal. Vertical coordination involves communication via supervisors and line managers whereas horizontal coordination is by one-to-one inter unit communication between actors in a non-hierarchical relationship (Lawrence & Lorsch, 1967).

In addition to defining various coordination modes, organisation theorists Van de Ven et al. (1976) also identified task uncertainty, task interdependence, and size of the work unit, as fundamental determinants of coordination mode. They have defined four workflow categories: independent,
sequential, reciprocal, and team. Each category evidences a different level of interdependence and an associated coordination mode. For example, team mode is shown to have the highest level of workflow interdependence, and is coordinated by group coordination using scheduled and unscheduled meetings.

Organisation structure and coordination mechanisms are linked in theoretical work by Mintzberg (1980) who defined five pure types of organisation structure and their dominating coordination mechanism. Mintzberg’s five structures include simple, machine bureaucracies, professional bureaucracies, the divisional form, and the adhocracy. Adhocracies, for instance, are comprised of multidisciplinary teams who treat problems as unique challenges requiring creative solutions and who fuse planning, design and execution work within projects. An adhocracy is characterised by mutual adjustment whereby individuals coordinate their own work by informal communication.

The practices of the published agile methods and the agile manifesto (Beck, et al., 2001) indicate agile approaches achieve coordination by feedback, mutual adjustment, group mode (formal and informal meetings), and personal horizontal coordination (Van de Ven, et al., 1976). Secondly, ‘team’ workflow fits with the self-organising team approach recommended in the agile manifesto (Beck, et al., 2001). Thirdly, an organisation using an agile approach aligns with an ‘adhocracy’ organisational structure, as proposed by Mintzberg. Finally, an agile approach is likely to engender high coordination cost (Van de Ven, et al., 1976) due to its reliance on mutual adjustment and group mechanisms.

Two issues arise when applying organisation theory to agile software development or any other type of IS project. First, organisation-level coordination research is based on theoretical and empirical work carried out in the pre-1990 era and the organisational IS environment of 2011 may not be comparable. Second, using organisation theories to explain and predict agile approaches assumes a project is a form of time-bounded organisation. While some researchers have made this assumption - for example, Barki, Rivard and Talbot (2001), Nidumolu (1995), and Xia and Lee (2005) - such an assumption is questionable. Organisation level theories may not hold at the project level due to the effect of time constraints that may not be present in permanent organisations (Barki, et al., 2001).

The information systems (IS) project is the second context in which coordination is a focus. Nidumolu (1995) investigated the impact of project uncertainty and vertical and horizontal coordination mechanisms on project performance in 64 IT projects, and found that horizontal coordination had a positive effect on project performance. Andres and Zmud (2001) studied the impact of task interdependence, coordination strategy, and goal conflict on software development success using an experimental approach, and found that organic coordination was generally superior in supporting success under conditions of high interdependence between organisational subunits. These results, from contexts other than agile software development, hint that agile software projects, which fit the theoretical organic and horizontal coordination strategies, would tend to be more successful.

Research into the impact of coordination within IS development on project success is diverse. A variety of contexts have been studied (e.g. IT projects, software projects, student projects, software teams) and a variety of ‘success’ constructs have been used (e.g. project satisfaction, productivity, process success, team effectiveness) (Aladwani, 2002; Andres & Zmud, 2001; Chen, Jiang, Klein, & Chen, 2009; Kraut & Streeter, 1995; MacCormack, Verganti, & Iansiti, 2001; Nidumolu, 1995; Nidumolu & Subramani, 2004). Overall, this body of research indicates that coordination is necessary, but not sufficient, for the success of software projects (Curtis, et al., 1988; Espinosa, Lerch, & Kraut, 2002; Kraut & Streeter, 1995; Nidumolu, 1995).

Within software projects, McChesney and Gallagher (2004) identified another important aspect of coordination; the presence of both explicit and implicit coordination. In their interpretive interview-based study of coordination practices in software engineering projects, they found that to explain all of the types of coordination mechanism in the two projects they studied, they needed to use, not only Coordination Theory (Malone & Crowston, 1994) but also collective mind theory (Weick & Roberts, 1993). They found that coordination has both explicit as well as implicit components, and they proposed that both types are necessary for a project to be successful.
Teamwork is the third context in which coordination is a focus. Teamwork research has identified an implicit form of coordination that goes by many different names. ‘Shared cognition’ is implicit coordination that occurs within work groups without explicit speech or message passing (Rico, Sanchez-Manzanares, Gil, & Gibson, 2008). Implicit coordination can occur via shared mental model (Kang, Yang, & Rowley, 2006; Yuan, Vogel, Zhang, Chen, & Chu, 2007), collective mind (Weick & Roberts, 1993), expertise coordination (Faraj & Sproull, 2000), and team mental model (Mohammed, Ferzandi, & Hamilton, 2010). Furthermore, Fiore and Salas (2004) specifically link team cognition and team coordination. Although these concepts are all slightly different, taken together they indicate that there is more to coordination than the explicit forms referred to in Coordination Theory (Malone & Crowston, 1994) and in organisation theory.

In summary, no single coordination theory explains or predicts the effectiveness of agile software development. This review has shown that Coordination Theory is not suitable for prediction although it is useful for identifying dependencies and coordination mechanisms. Furthermore, the three coordination contexts relevant to a study of agile software development; organisational, IS project, and teamwork, provide many coordination concepts that could be applied to the context of agile software development but no empirical work could be found linking agile software development with these concepts. In addition, coordination modes identified in organisation level theories, such as mutual adjustment and adhocracies, may not be relevant today, nor applicable at the project level. Finally, IS coordination research has identified that coordination is necessary but not sufficient for project success, and teamwork studies have shown that implicit forms of coordination exist in addition to the explicit forms identified in organisation studies and Coordination Theory.

4 CONCEPTUAL MODEL

Providing clear definitions of concepts or constructs is necessary in any theory building exercise (Suddaby, 2010). The aim of this paper is to develop a concise definition for the concept of project coordination effectiveness. To begin to define a concept of coordination effectiveness in the context of agile software development the work of Espinosa et al. (2002) was drawn on. Espinosa et al. (2002) melded much of the previous coordination research work, discussed in the literature review, into a single framework linking dependencies, explicit, and implicit coordination mechanisms. In that framework, they distinguished between two coordination concepts: “coordinating” and “state of coordination.” Coordinating is the managing of dependencies; it is the combination of different coordination mechanisms used within a project to achieve coordination. In contrast, the state of coordination is the extent to which that coordination is effective.

In this paper, these two concepts are renamed to clarify the distinction between them. Coordinating is renamed coordination strategy, which is envisaged to be a group of coordination mechanisms that occur in agile software development projects. State of coordination is renamed project coordination effectiveness. The focus in this paper is on defining project coordination effectiveness.

The conceptual framework for this research study is as shown in Figure 1. The overarching concern is the influence of the coordination strategy on coordination effectiveness in agile software development projects. The particular focus of this paper is on coordination effectiveness (shown in black in Figure 1). The definition of coordination strategy, and the relationship between strategy and coordination effectiveness, project size, complexity and uncertainty, will be examined in future research. Further evidence from case studies will be used to explore these concepts and relationships.
The initial concept of coordination effectiveness has two components; an explicit part comprising overt physical actions and artefacts, and an implicit, or knowledge-based part that involves ‘knowing’ things about the project.

The explicit component involves the objects (persons or artefacts) involved in a project. A project is coordinated effectively when the required object is in the correct place at the correct time and in a state of readiness for use from the perspective of each individual involved in the project. In common parlance: the right thing is in the right place at the right time. This principle is drawn from Coordination Theory (Malone & Crowston, 1994) and Crowston’s (2003) taxonomy relating generic dependencies and coordination mechanisms.

The implicit component comprises knowledge held by project team members, and is drawn from the teamwork literature about team, or shared, mental models (Mohammed, et al., 2010). This research focuses on teams working in environments characterised by complexity and change, where effective teams have shared knowledge about “what is happening, what is likely to happen next, and why it is happening” (Mohammed, et al., 2010, p. 879). Implicit coordination is also concerned with knowing who has the needed skill and expertise to complete tasks (Faraj & Sproull, 2000).

5 RESEARCH METHOD

To develop a concept of coordination effectiveness based on empirical data from software development projects the positivist multi-case study research method was selected. This is a common and well-defined way to investigate phenomena in natural information system development contexts where events cannot be controlled and where it is important to capture the detail in a situation (Darke, Shanks, & Broadbent, 1998; Eisenhardt, 1989; Pare, 2004; Yin, 2003). Furthermore, this method is suitable for building theory, which is the aim of the research (Eisenhardt & Graebner, 2007). The guidelines of Dube and Pare (2003) for carrying out rigorous exploratory positivist case study research were followed in designing the study and performing data collection and data analysis.

5.1 Case selection strategy

The unit of analysis, or case, was a software development project with co-located participants. Case selection followed a replication logic strategy as recommended by Miles and Huberman (1994). This involves selecting cases that tend to either confirm (literal replication) or disconfirm (theoretical replication) the theory of interest.

Four projects were selected. Three were literal replications because they were using the common agile software development methods, Scrum, or Scrum with XP. One further project was a theoretical replication because traditional project management was used with no particular system development methodology. In this way, case variability was maximised by selecting ‘polar types’ as recommended by Eisenhardt and Graebner (2007) to provide better concept definition by accounting for a variety of
contexts and factors in the emergent theoretical concept. To maximise variability, projects were selected from different organisation types and involving different types of development contract.

All of the projects were conducted in Wellington, the capital city of New Zealand (NZ). Wellington is a centre for software development because government departments and company head offices are located there. To distinguish this research from distributed agile software projects, all projects were required to have co-located teams. Selected projects had to be at least one-half completed or completed within one month prior to the interviews so that projects were well underway and participants were recalling current events. Projects had to include a team of at least two people, the software under development had to be at least moderately complex, and be of at least moderate importance to the client or host organisation. The researcher in consultation with a key informant assessed these aspects subjectively. Complexity was assessed as a combination of team size, number of stakeholders, number of integrating systems, and complexity of requirements.

Cases were located through a not-for-profit group, the Agile Professionals Network (APN), and from the researchers’ extensive professional network. The aim of APN is to inform organisations about the agile approach by hosting monthly seminars on agile adoption in organisations. Contacts within APN and speakers at seminars were approached to enquire if they knew of appropriate projects. If this was so, a meeting was held with a key organisational contact to see if the project met the selection criteria. Not all people approached agreed to participate, and initial contacts were discontinued if their project did not meet the selection criteria. In this way, four projects were selected and code-named, Hour, Land, Storm, and Silver. Table 1 shows the project profiles.

<table>
<thead>
<tr>
<th>Organisation type</th>
<th>Hour</th>
<th>Land</th>
<th>Storm</th>
<th>Silver</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not-for-profit</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Government</td>
<td></td>
<td></td>
<td>Commercial service provider</td>
<td></td>
</tr>
<tr>
<td>To comply with new government regulations</td>
<td>To improve the organisations interactions with the public</td>
<td>To migrate a critical legacy system to a modern technology platform</td>
<td>To provide a new reporting system for an external client</td>
<td></td>
</tr>
<tr>
<td>In-house development</td>
<td>In-house development</td>
<td>Independent contractors working on the client site</td>
<td>Development for external client</td>
<td></td>
</tr>
<tr>
<td>Non-agile</td>
<td>Scrum</td>
<td>Scrum and XP</td>
<td>Scrum</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>10</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

Table 1. Case and interviewee profiles.

5.2 Data collection

The first step in data collection was to develop an interview schedule based on the research on coordination in software projects and IT projects. The interview schedule was based on those of Crowston (1991) and McChesney and Gallagher (2004). Spradley’s (1979) advice on interviewing in natural situations was followed. In particular, Spradley suggests avoiding ‘why’ questions to ensure that participants gave a full and open range of responses about coordination and agile practices within the project. The primary data collection method was semi-structured interview of software project team members. An initial interview was conducted with one senior team member who could provide details on the project, its purpose, background, history, and issues. This person was often a project
manager, but not in all cases, as not all teams included a formal project manager role. In each project, up to four other project team members including developers, business analysts, domain experts, and testers, were interviewed for 1 to 1½ hours. All interviews were voluntary. As well as interview transcripts, source data included observations made while visiting work sites and observing meetings, project documents, and photographs taken at the work sites.

In each interview, three specific questions, designed to elicit information about project effectiveness, were asked:

1. What makes this project well-coordinated?
2. What interferes with coordination on this project?
3. In your opinion, based on all of the software development projects you have worked on, what is a well-coordinated project?

Thirteen individuals were interviewed and each interview was digitally recorded with the permission of the participant. Participants were informed before the interview that all data was to be confidential.

5.3 Data analysis

Data analysis involved first transcribing all of the interview recordings in full and then uploading the textual data into the qualitative data analysis tool NVivo. Data was pooled across the four cases, that is, no within-case analysis was performed. Next, the text from each interview was analysed in turn. Each transcript was reviewed and coded following the coding approach proposed by Miles and Huberman (1994). A data unit was a sentence, and sometimes a short paragraph depending on how much conversational text conveyed the participants meaning. Each data unit was allocated to one of the two groups identified in the initial conceptual model, that is, implicit and explicit aspects of coordination effectiveness. Explicit coordination codes were predefined as ‘right place’, ‘right time’, and ‘right thing’; implicit codes were not initially subdivided. However, during the analysis process, implicit coordination elements were subdivided into separate codes in instances where it was clear that participants were reporting more than one type of implicit coordination. The codes are shown in Appendix 1, Tables 2 and 3 in the leftmost column of each table.

6 FINDINGS

The final definition of agile project coordination effectiveness was developed from the case data and the literature on coordination. The case analysis found evidence for both explicit and implicit components of project coordination effectiveness (Appendix 1 shows samples of evidence from the 13 participants).

The research participants identified each of the three proposed explicit components of coordination, ‘right thing’, ‘right place’, and ‘right time’. Evidence for these components is provided in Appendix 1 Table 2.

The research participants identified four implicit components of coordination including ‘Know why’, ‘Know what is going on and when’, ‘Know what to do and when’, and ‘Know who is doing what’.

The ‘know why’ component is about each individual working on the project understanding the overall project goal and understanding how a task contributes to that overall goal. As one software developer explained: “You know enough that you know what you are a part of, rather than being told to work on a little corner of something and never knowing what you are actually doing it for.” [SD Storm].

The ‘know what is going on and when’ component is about each individual working on the project having an overall idea about the project status, that is, tasks that are currently underway and tasks that need to be performed in the future. Evidence for this component is shown in Appendix 1 Table 3.

The ‘know what to do and when’ component is about each individual working on the project knowing what task they should be working on and when they should be working on that task relative to all of the other tasks that must be completed. Evidence for this component is shown in Appendix 1 Table 3.
The ‘know who is doing what’ component is about each individual on the project knowing what tasks others are currently working on. Evidence for this component is shown in Appendix 1 Table 3.

An implicit coordination component was taken from the work by Faraj and Sproull (2000) on expertise coordination in software teams. They identified three components of expertise coordination, knowing where expertise is located, knowing where expertise is needed (both are implicit aspects of coordination) and bringing needed expertise to bear (an explicit aspect of coordination, which is an example of the ‘right thing’ coordination component identified in this paper). Although there was no direct evidence for expertise coordination in the findings from this study, Faraj and Sproull’s arguments and findings are compelling. Therefore, ‘Know who knows what’ was added to the definition. The final definition for coordination effectiveness developed from combining the interview data, and the literature on expertise coordination is illustrated in Figure 2.

### Figure 2. Components of coordination effectiveness.

The written definition of coordination effectiveness in this context is as follows:

*Coordination effectiveness is a state of coordination wherein the entire agile software development team has a comprehensive understanding of, the project goal, the project priorities, what is going on and when, what they as individuals need to do and when, who is doing what, and how each individuals work fits in with other team members work. In addition, every object (thing or resource) needed to meet a project goal is in the correct place or location at the correct time and in a state of readiness for use from the perspective of each individual involved in the project.*

The difference in the perception of coordination effectiveness between the agile and non-agile participants is interesting. The agile participants gave ‘team’ focused responses often using the word ‘team’ and ‘everyone’ in their responses, whereas the non-agile participants assumed that coordination is achieved by one central controller or ‘key person’ who will guide and inform all of the other project stakeholders. These responses came from the non-agile project, Hour. In that project, the project manager spoke of the project members as being a ‘team’, but the individuals working on the project never mentioned ‘team’ nor indicated that this concept was important in their perception of coordination effectiveness. There is not enough evidence to state this finding strongly as project Hour was quite small, however this difference appears to be worthy of further investigation.

### 7 CONCLUSION

The aim of the study was to define a concept of project coordination effectiveness in the context of agile software projects. In the tradition of theory building from positivist case study research as discussed by Eisenhardt and Graebner (2007), a theoretical concept of coordination effectiveness was developed based upon data from four cases and from extant literature on software project
coordination. The qualitative data drew from 13 participants in three agile software development projects and one non-agile project.

The study has limitations. The number of participants was small; the findings would be strengthened with a larger sample allowing for theoretical saturation (Miles & Huberman, 1994). Also, case selection, and therefore the findings, are based upon the responses of co-located teams; the validity of these findings in the case of distributed teams has not been established.

Another limitation of this study is that individual with-case analysis was not carried out, rather, the data from all cases was pooled. This means that individual project factors, such as project complexity, uncertainty, or size that might influence the participant’s responses were not investigated.

When considering the applicability of the findings in a wider context, it is worth noting that although this research was conducted in the New Zealand context, at least one third of the research participants came from other countries where they were trained before immigrating to New Zealand. Therefore, the findings may be applicable internationally in projects with a similar profile to those in this study, since the experiences and training of the participants is likely to be reflected in their responses to the interview questions.

The grounded definition of coordination effectiveness developed in this paper makes possible a more nuanced picture of how coordination is achieved in agile software development projects in future work. One path forward would be to apply the definition so as to create an operationalised construct, suitable for testing using a quantitative method. This would enable a study of the coordination mechanisms (e.g. certain agile practices) and dependencies occurring in agile software development projects, and their affect on coordination effectiveness. The concept might also be used to compare coordination between software projects, and between different types of software project, for example agile and non-agile projects, and between different system development methodologies. The concept may also be useful for researchers studying software teams, and in non-software contexts such as general teamwork, project work, and R&D work.

A well-formed concept of agile software development project coordination effectiveness provides a useful tool for extending research in this underexplored area. Agile activities and artefacts that support coordination can now be investigated, and coordination theorists may choose to use this concept to gain interesting new insights into coordination in other contexts.
Appendix 1

<table>
<thead>
<tr>
<th>Explicit Coordination</th>
<th>Sample Evidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Right thing</td>
<td>“You know there was a decision to be made, but they couldn’t make it without knowing the technical implications. So they wanted to wait for me to get back. So that probably interfered with coordination, not having me present all the time.” SD Land</td>
</tr>
<tr>
<td>Right place</td>
<td>“Accessibility of people [all of the stakeholders]” SD Land</td>
</tr>
<tr>
<td>Right time</td>
<td>“As long as these people were accessible to Mark, the communication would be flowing, but if these people were not accessible, or there was a time lag, then coordination becomes problematic.” SD Land</td>
</tr>
<tr>
<td></td>
<td>“You want someone to be able to react at the time something happens.” PM Storm</td>
</tr>
<tr>
<td></td>
<td>“It [user stories] allowed a cohesive flow of the development so we didn’t end up developing something that couldn’t be immediately integrated into the project.” SD Silver</td>
</tr>
</tbody>
</table>

Key for Tables 1 and 2
PM - Project manager
SD - Software developer
DE – Domain expert
T – Tester
AC – Agile coach

Table 2. Coordination effectiveness - evidence for explicit coordination.

<table>
<thead>
<tr>
<th>Implicit Coordination</th>
<th>Sample Evidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Know why - team members have a shared understanding of the project goal and the project priorities</td>
<td>“I would think of coordination as being everyone working together towards the same goal and understanding what people’s roles are within that.” SD Hour</td>
</tr>
<tr>
<td></td>
<td>“No one is able to say ‘this one is more important than that one, so stop working on that one’. There are different groups of people all saying ‘this is the top priority’. And no one able to say, ‘Ok, you have got three things of equal priority; in fact they are in this order of 1, 2, and 3. No one able to make that decision, or get agreement on those relative priorities, and so you are left in a position of three equally important high priority things...So that was the obstacle to the coordinating thing.” PM Land</td>
</tr>
<tr>
<td></td>
<td>“...just being in the situation where you have a good understanding... you know enough that you know what you are a part of, rather than being told to work on a little corner of something and never knowing what you are actually doing it for.” SD Storm</td>
</tr>
<tr>
<td></td>
<td>“To me, coordination is basically running in the same direction, meaning there is a shared goal and people know what they are supposed to do.” AC Silver</td>
</tr>
<tr>
<td>Know what is going on - everyone on the team knows what is</td>
<td>“If you have a key person who pushes, not pushes, but who keeps the momentum going and sends out all the reporting, and keeps everyone updated, and makes sure everyone knows what they are working on and what they should be doing.” SD Hour</td>
</tr>
<tr>
<td></td>
<td>“because I think the result of that was that decisions took longer than they should have. So when they were talking about, ‘Ok well”</td>
</tr>
<tr>
<td>Implicit Coordination</td>
<td>Sample Evidence</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------</td>
</tr>
</tbody>
</table>
| going on and when within the project | **what is the real, you know, what is this part of the system going to look like’**, well it depends on what those guys of doing, and what these guys are doing.” SD Land  
  - “So those things [sprints, stand-ups, retrospectives, planning sessions] were the bread and butter that meant we knew the things that we were doing.” SD Silver |
| Know what to do - everyone on the team knows what they need to do and when | **“Everyone knows what is expected of them and when and what the impact is if they don’t do that, how that is going to impact on other people.”** PM Land  
  - “…that was really important to just make sure everyone was on the same page and no one was drifting too much” SD Land  
  - “My team, I can tell if they are well co-ordinated by the fact that they are not confused about what they have to do each day I guess, if they know what they are doing and what they have to do next; then they are well co-ordinated.” PM Storm  
  - “Because I know what to do. Each time I don’t get lost, and I, even having a lack of information because of my English condition, I can know what I have to do, what people expect me to do.” SD1 Storm  
  - “And my guys on my team think it’s well coordinated when they know exactly what the next bit of work they have got to do is” PM Storm  
  - “I guess from a developer point of view, just being in the situation where you have a good understanding of everything that you need to do.” SD2 Storm  
  - “Or an agreed upon process. I have definitely worked in smaller companies, where no one has actually gone to the trouble of spelling out the process, it has just been assumed…you know, what are we gonna, ‘how are we going to capture requirements’, ‘what are we considering “done”’. You know like a lot of that stuff, in a lot of organisations, a process is not really clearly defined. No one has gone to the trouble of saying, all right, well, when you have done this, then we can say that this is done, and it will go to the next stage.” SD Land |
| Know who - everyone on the team knows who is doing what and how their work fits with other peoples work | **“and we saw that with this project when we got comments like, ‘did not want to let the team down’ that was an understanding of the impact that not delivering on something was going to do to the rest of team and therefore on our ability to get that end game of being seen to be successful.”** PM Land  
  - “I guess from a developer point of view, just being in the situation where you have a good understanding of everything that you need to do. Everything that is happening; everything that is happening around it.” SD Storm  
  - “…people need to find out who is doing what in a flexible way, and in a way that they decide for themselves, so they own this process” AC Silver |

*Table 3. Coordination effectiveness – evidence for implicit coordination.*
References


