Chapter 4

A Coordination Perspective on Agile Software Development

Diane E. Strode
Whitireia Polytechnic, New Zealand

Sid L. Huff
Ivey Business School, Canada

ABSTRACT

Achieving success in software development projects is a perennial challenge, and agile software development methods emerged to tackle this challenge. Agile software development provides a way to organise complex multi-participant software development projects while achieving fast delivery of quality software, meeting customer requirements, and coping effectively with project change. There is little understanding, however, of how such projects achieve effective coordination, which is a critical factor in successful software projects. Based on evidence from four cases, this chapter presents a theory explaining coordination in agile software projects. This theory defines the concepts of coordination strategy and coordination effectiveness and propositions explaining their relationship. This theory contributes to coordination literature by presenting clearly delineated concepts and their relationships in the form of a variance theory. For IT project management, this theory contributes to knowledge of coordination and coordination effectiveness in the context of agile software development.

INTRODUCTION

How can software development projects be coordinated more effectively in the presence of uncertainty? (Nidumolu, 1995, p. 213)

Agile software development offers an effective means of organising complex multi-participant software development while continuously adjusting to project uncertainty and change. While this approach is now widely accepted, it remains of intense practical and theoretical interest because of its impact on people, projects, and organisations (Ambler, 2009; Dyba & Dingsoyr, 2008; Strode, Huff, Hope, & Link, 2012; West & Grant, 2010). Agile software development differs significantly from earlier approaches to software development; because of this difference, adopting an agile approach can affect organisation structure and culture, project roles, contract negotiation, the
way customers are involved in projects, system quality, and system deployment. In addition, the agile approach changes technical practices, team social interaction, team leadership, and decision-making. These wide-ranging effects mean that existing theories about software development organisations, project management, and system development need to be re-evaluated in light of this new systems development paradigm (Dingsoyr, Dyba, & Moe, 2010). This chapter contributes to this re-evaluation by presenting a theory of agile software development from a coordination perspective.

Long before the advent of agile software development, researchers identified effective coordination as a critical element in organisations generally, and in software development projects in particular (Curtis, Krasner, & Iscoe, 1988; Kraut & Streeter, 1995; Van de Ven, Delbecq, & Koenig, 1976). Nidumolu (1995) articulated the problem when he asked how software development projects could be effectively coordinated in the presence of uncertainty. Shortly thereafter, software developers and systems methodologists designed their agile methods specifically to deal with change and uncertainty. These methods de-emphasize traditional coordination mechanisms such as forward planning, extensive documentation, detailed contracts, strict adherence to a pre-specified process, and specific coordination roles such as that of the project manager. Instead, they favour intensive face-to-face communication and other apparently simple practices. Yet, agile software development has contributed to the success of many software projects, so arguably effective coordination of some type takes place within these projects (Agerfalk, Fitzgerald, & Slaughter, 2009; Cao & Ramesh, 2007; Dingsoyr, Dyba, & Abrahamsson, 2008; Kautz, Madsen, & Norbjerg, 2007).

A large and diverse body of coordination research exists in contexts relevant to software development practice, but agile software development came into common use subsequent to much of that research. Therefore, we do not know whether existing theory adequately explains coordination within agile software development projects, or whether a new explanatory theory is needed.

In studying coordination, there are two primary considerations. The first is to identify a coordination strategy, that is, those activities and artefacts in a situation that support coordinated action. The second is to identify the characteristics of a highly coordinated state. Once we understand these two aspects of coordination, it becomes possible to investigate which activities and artefacts are more, or less, effective at bringing about a highly coordinated state. Consequently, this study is guided by two research questions: How is coordination achieved when using an agile software development approach? And what is the relationship between coordination strategy and project coordination effectiveness in the context of agile software development?

Following the guidelines of Eisenhardt and Graebner (2007) for theory building from positivist multi-case studies, in this chapter we develop a theory of agile software development project coordination by drawing on evidence from three independent cases of agile software development. A single contrasting case where the project was carried out with a non-agile approach is included to illustrate the boundaries of the theory. The case evidence shows that many practices in the agile projects act as coordination mechanisms. Together, these mechanisms form a project’s coordination strategy, which has three components: synchronization, structure, and boundary spanning. The cases also show that an agile coordination strategy leads to coordination effectiveness, which has both implicit and explicit components. We elaborate the relationships between these concepts in a series of propositions.

The chapter is organised as follows. First, we review the current state of research on agile software development. Then we discuss coordination research in organisation studies, information systems project studies, and teamwork. Then we