December 2006

Coordination in Agile Software Projects

Peng Xu
University of Massachusetts Boston

Lan Cao
Old Dominion University

Coordination in Agile Software Projects

Peng Xu
Department of Management Science and Information Systems
University of Massachusetts Boston
Email: peng.xu@umb.edu

Lan Cao
Department of Information Technology and Decision Sciences
Old Dominion University
Email: lcao@odu.edu

ABSTRACT

Agile methodologies have been widely adopted by various projects. Though different practices are recommended by different methodologies, they all share some common key characteristics. These characteristics call for more effective coordination within development teams. In this research, we investigate the coordination mechanisms used in agile software development and the factors that impact their effectiveness. We differentiate coordination mechanisms as vertical coordination and horizontal coordination. We argue that vertical coordination and horizontal coordination are positively associated with the success of agile projects. The relationship between vertical coordination and the success of agile project is moderated by project size. Explicit knowledge representation can positively influence vertical coordination. Sharing locus of practices can increase collective identity, both of which can positively influence horizontal coordination.

KEYWORDS:

Coordination, Agile methodology, Vertical coordination, Horizontal coordination

INTRODUCTION

Various agile methods (e.g., eXtreme Programming, Scrum, and Crystal Clear) have been proposed and widely adopted in software development projects. Although there are several agile methods focusing on different perspectives in software development, a dominant idea in agile methods is that the development team can be more effective in responding to changes if it can improve the coordination between people (Cockburn and Highsmith, 2001). The common characteristics shared by various agile methods such as collaborative work, incremental evolutionary life cycle, and strong customer communication (Levine, et al., 2002) call for effective coordination mechanisms that are different from ones used in traditional development. Agile methods believe that face-to-face conversation is the most efficient and effective method of conveying information. The methods focus on fostering a high degree of interaction among team members and the project's customers. Based on this belief, heavy documentation, upfront design, detail project plans and formal contracts are replaced by pair programming, evolving design, planning game and co-located customers. Rather than having workers in specialized roles handing off documentation to workers in other specialized roles, agile practices more emphasize on inter-personal information and knowledge sharing.

Effective coordination is critical for agile methods. Though agile methods propose the practices that can be followed during development, the guidelines on coordination in agile projects are missing. Prior research on coordination modes mainly focuses on traditional hierarchical business setting or fairly stable business environment (Malone and Crowston 1994; Van de Ven, et al., 1976). Few studies have been done to understand coordination modes and strategies used in agile software development. Software projects adopting agile methods face coordination challenges such as how to balance the structure and agility when choosing coordination mechanisms, and how to adjust coordination mechanisms when the project scales up since most agile practices are mainly for small size projects. Motivated by this observation, in our research, we will investigate the research questions:

1. What are the impacts of different coordination mechanisms on agile project performance?
2. What are the factors that impact the effectiveness of a coordination mechanism in agile software development?

COORDINATION IN AGILE SOFTWARE DEVELOPMENT

Coordination is defined as "integrating or linking together different parts of an organization to accomplish a collective set of tasks" (Van de Ven, et al., 1976). Coordination process can help organizations or organizational units to recognize the valuable resources, allocated resources effectively, assemble resources to execute tasks, and accomplish new tasks in dynamic contexts (Malone and Crowston, 1994; Quinn and J.E. Dutton, 2005).

Coordination has been considered as one of the most important factors that influence software project development (Faraj and Sproull, 2000; Hardy, et al., 2005; Levina, 2005). Various coordination mechanisms have been used in IS projects such as formalization of project management (Kiesler, et al., 1994) and hierarchical and lateral direct communications between interdependent actors (Nidumolu, 1995; Thompson, 1967). Project management can take means of specifying task decomposition, authority structures, work planning and standard operating procedures (Kiesler, et al., 1994). Hierarchical and lateral communications include formal communications that handle routine work and informal communications that address unpredicted events (Kraut and Streeter, 1995).

In agile software projects, coordination processes face similar challenges as traditional software project such as allocating resources, assigning tasks, and maintaining the relationships within teams. However, the focuses of coordination in agile projects are different from the traditional approaches. Agile methods try to minimize the overheads associated with formal procedures and increase process agility by emphasizing inter-personal coordination. The coordination mechanisms in agile practices can be classified as horizontal coordination and vertical coordination based on the channel they use.

- Vertical coordination

Vertical coordination involves more formal channel than horizontal coordination. It is done by involving supervisors and other relatively formal and/or centralized control mechanisms. In our study, vertical coordination refer to the coordination process where agile development team members coordinate via task management that specifies work inputs, behavior or outputs (Van de Ven, et al., 1976) and hierarchical structures that define authority relationships (Weber, 1946). In agile software development, task management takes the form of continuous planning, task assignment, essential documentations such as user stories, iteration and release control. These task management activities and documents reflect the decisions made by the authority (e.g., managers) and serve as the norms and guidance on how the project should precede. Team members need to refer to the guidance for their activities. Therefore, vertical coordination can be done via task management in agile projects. Hierarchical structures defines how the work is accomplished and managed by the different roles in agile project such as coach, project leader, team leader, managers, and product managers. In agile software development, problems and conflicts are emerging constantly; activities and plans need to be modified to respond to changes in a timely manner. Any coordination that involves the hierarchical structure for facilitation and/or direction falls in the vertical coordination category.

- Horizontal coordination

Contrary to vertical coordination, horizontal coordination mainly involves informal, peer-to-peer interaction. Derived from the concept of direct communications between interdependent actors (Nidumolu, 1995; Thompson, 1967), horizontal coordination in this study is defined as the laterally direct communication and cooperation between team members. One significant difference between agile methods and traditional methods is the emphasis on horizontal coordination in software development. In agile projects, intensive communications and direct interaction among team members and between the team and customers are the key factors. Customers are involved in the development process as a team member. Co-located developers frequently talk to each other face-to-face to solve the problems occurred. Collective ownership of the code and trust fostered with the team enable developers to be able to make timely decisions without reporting to higher authority in some occasions.

In this study, we investigate the roles played by these two different coordination strategies in agile development, how they influence the performance of agile development, how they are influenced by other factors.

RESEARCH MODEL

Figure 1 describes our research model.

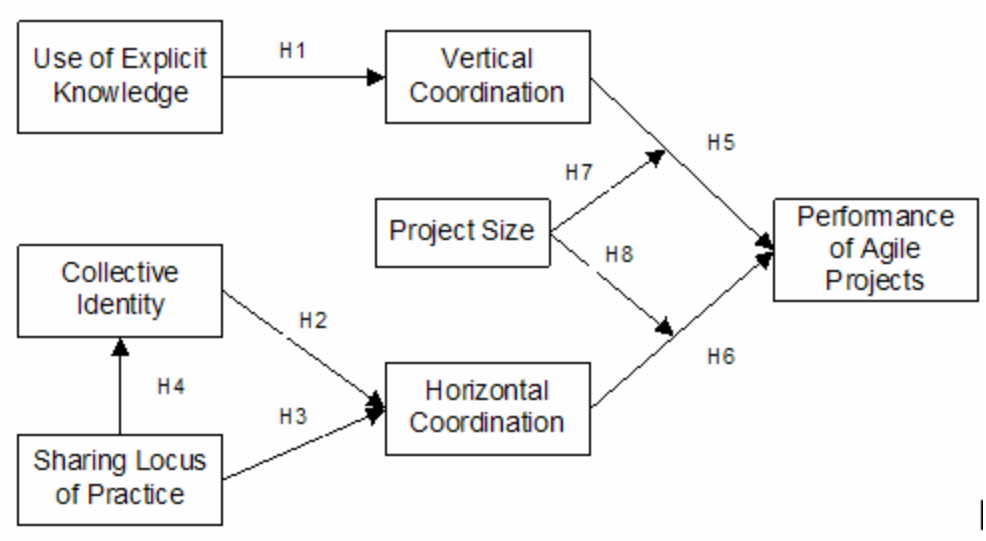

Figure 1. Research Model

Use of Explicit Knowledge

In software development, various stakeholders with different backgrounds are involved. Each stakeholder mainly focuses on her or his own area. Successful software projects need to overcome the differences between stakeholders and effectively integrate expertise to successfully deliver products (Tiwana and Mclean, 2005).

An effective way to overcome differences and coordinate in product development is to codify knowledge into artifacts and documents that are shareable across different problem solving contexts (Carlile, 2002; Carlile, 2004). Explicit and sharable knowledge include shared definitions across boundaries, forms and methods that provide a shared format for solving problems, objects or models that represent problems or solutions, and maps of boundaries that represent the dependencies and boundaries. These knowledge has been considered as one of the mechanisms to share and transform knowledge between given boundary such as different experts, teams, departments and organizations in product development (Bechky, 2003; Carlile, 2004; Levina, 2005).. It helps project managers gather, organize, and represent information of the project, which can be used to monitor progress, measure the status, allocate resources and facilitate decision-making.

Though agile software development mainly relies on sharing tacit knowledge to increase flexibility and remove overheads, essential knowledge such as user stories and iteration plans needs to be explicitly documented for control and communication purposes. User stories help document user requirements and estimate development efforts. Plans can measure the project status. Prior research also suggests that relatively large, complex projects need more documents such as intermediate design documents to explicitly document project knowledge and increase control and coordinate development effort by project mangers (Schalliol, 2001). It shows that the lack of explicit knowledge can causes difficulties in vertical coordination.

Therefore, we argue that though agile methods significantly reduce the number of documents and artifacts, necessary explicit knowledge should be used to support vertical coordination.

H1. The explicit knowledge representation in agile software development positively affects vertical coordination.

Collective Identity

Collective identity stresses shared attributes and beliefs among team members, provides a sense of belonging and helps teams collaborate (Levina, 2005). Collective identity gives an identity that is meaningful to team members who can collectively engage in the discursive practices (Hardy, et al., 2005). Collective identity is a valuable resource possessed by agile development teams that can help horizontal coordination between team members. In agile software development, various stakeholders interact with each other on daily basis. Their concerns and interests vary. Stakeholders need to share and

negotiate with one another during the development process in which common knowledge and goals play an important role (Bechky, 2003; Levina, 2005). With high degree of collective identity, everyone shares common goals and values. These shared attributes will facilitate knowledge sharing, reaching consensus and even making compromises if necessary.

Several agile practices actually are proposed to strengthen team identity. For example, Extreme programming employs pair programming where two programmers work on the same piece of code at the same time and collective code ownership where everyone of team owns the whole system.

Therefore we argue that in agile development, the high level of collective identity will facilitate horizontal coordination.

H2: Collective identity positively affects horizontal coordination.

Sharing Locus of Practices

Locus of one's practices refer to the main attention of one's job or the core nature of one' work (Bechky, 2003). Each participant on development teams has his or her own locus of practice. For example, product managers are concerned with user satisfaction and delivery time, while test engineers are concerned with quality. Different locus of practices can invoke two problems: miscommunication/knowledge misinterpretation that will hinder horizontal coordination and different interests that will lower collective identity.

In agile development, most of knowledge about the project is localized, embedded and invested in practice. With the nature of tacit knowledge and diverse backgrounds of stakeholders, consequences of knowledge sharing depend on interpretation of knowledge. Different locus of practices of team members will lead to little common knowledge and language shared by team members, which tends to lead to less knowledge sharing and more misinterpretation of knowledge (Carlile, 2004).

To overcome the challenges, agile development needs to improve common understanding among team members. An effective way to achieve this goal is sharing locus of practice of each participant such as rotating pairs and tasks among team members, and creating standards across team (e.g., coding standard in XP). Doing so can build up common ground of team members, help them to learn and transform their own knowledge, and remove misinterpretation of other tacit knowledge. Doing so can facilitate peer communication, interaction and cooperation, thus improving horizontal coordination. Do so can also cultivate team culture and build trust with the teams, thus increasing team members' collective identifies.

H3: Sharing locus of practice positively affects horizontal coordination.
H4: Sharing locus of practice positively affects collective identity.

Vertical Coordination

The importance of vertical coordination in software development has been recognized. Formal coordination mechanisms to assign tasks, allocate physical and economic resources, manage resource dependencies, and integrate outputs positively influence team performance (Faraj and Sproull, 2000). Individual efforts can be structured and synchronized by vertical coordination to improve teamwork (Hoegl and Gemuenden, 2001).

Though agile methods avoid unnecessary formal control overheads such as formal documents, meetings, and plans, they do not mean less or no vertical coordination. Instead, vertical coordination is very essential for the success of agile methods. The status of a project is continuously measured based on short iteration cycles. Project leaders constantly revise project plans and take quick actions to adjust processes to address unpredicted and ad-hoc concerns.

H5: Vertical coordination positively affects the success of agile projects performance.

Horizontal Coordination

Agile methods rely on collaborating with the customer over contract negotiations, responding to changes, frequent releases, and multiple iterations. The methods value collective reflection-in-action, i.e., using conversations among project participants who collaborate to leverage the differences among them to produce synergistic solutions and balances divergent stakeholders'

concerns (Levina, 2005). In software development, the coordination of the team shows a strong relationship with team performance (Faraj and Sproull, 2000). In agile development, due to the lack of formal communication channels (e.g., formal documents and meetings) and unpredictable process, the inter-personal coordination among team members and between teams and customers is more important than ever (Cockburn and Highsmith, 2001).

H6: Horizontal coordination positively affects agile projects performance.

Project Size

Though agile methods were argued as only for small projects (Boehm, 2003), more and more medium-sized or even large-sized projects are adopting these methods (Schalliol, 2001). When the project is small, close interaction between team personnel is easy. Problems can be quickly spotted and corrected. However, as the size of the project increases, the chances for close interaction with each other within the team drop (Van de Ven, et al. 1976). Miscommunication, misunderstanding and confusion are more often to happen and more difficult to identify. In this case, horizontal communication such as direct and information communication between team members give way to more impersonal coordination (Van de Ven, et al., 1976) as vertical coordination that involves high level of control can help organize and evaluate the project.

H7: Project size moderates the positive impact of vertical coordination on agile projects performance. The positive impact of vertical coordination of large projects on performance is more obvious than that of small projects.

H8: Project size moderates the positive impact of horizontal coordination on agile projects performance. The positive impact of horizontal coordination of small projects on performance is more obvious than that of large projects.

CONCLUSION AND FUTURE WORK

We will use survey methodology to empirically validate the research model. Questionnaires, which are under development, will be sent to agile practitioners.

We believe that our study will contribute to both theories and practices. Various agile methods that describe different practices and recommendations have been proposed. Different types of projects have tried to adopt agile practices. So far, the results are mixed. Limited research has been conducted to understand agile practices from theoretical perspective. Practitioners need more guidelines on what practices may be adopted and why. Our research focuses on coordination issues in agile projects and use coordination theories to explain the agile practices and their relationships with the success of agile projects.

REFERENCES:

1. Bechky, B.A. (2003) Sharing meaning across occupational communities: The transformation of understanding on a Production floor, *Organization Science*, 14,3, 312-330.
2. Boehm, B. (2003) Balancing agility and discipline: A guide for the perplexed, Addison Wesley Professional, Boston.
3. Carlile, P.R. (2002) A Pragmatic view of knowledge and boundaries: Boundary objects in new product development, *Organization Science*, 13,4, 442-455.
4. Carlile, P.R. (2004) Transferring translating, and transforming: An integrative framework for managing knowledge across boundaries, *Organization Science*, 15,5, 555 –568.
5. Cockburn, A., and Highsmith, J. (2001) Agile software development: The people factor, *IEEE Computer*, 34,11, 131-133.
6. Faraj, S., and Sproull, L. (2000) Coordinating expertise in software development teams, *Management Science*, 46,12, 1554-1568.
7. Hardy, C., Lawrence, T.B., and Grant, D. (2005) Discourse and collaboration: The role of conversations and collective identity, *Academy of Management Review*, 31,1, 58 –77.

8. Hoegl, M., and Gemuenden, H.G. (2001) Teamwork quality and the success of innovative projects: A theoretical concept and empirical evidence, *Organization Science*, 12,4, 435-449.
9. Kiesler, S., Wholey, D., and Carley, K.M. (1994) Coordination as linkage: The case of software development teams, In H. Harris (ed.) *Organizational Linkages: Understanding the Productivity Paradox*, 52, National Academy Press, Washington D.C., 96-123.
10. Kraut, R.E., and Streeter, L.A. (1995) Coordination in software development, *Communications of the ACM*, 38,3, 69-81.
11. Levina, N. (2005) Collaborating on multiparty information Systems development projects: A collective reflection-in-Action view, *Information Systems Research*, 16,2, 109 –130.
12. Levine, L., Baskerville, R., Loveland Link, J.L., Pries-Heje, J., Ramesh, B., and Slaughter, S. (2002) Discovery colloquium: Quality software development at Internet speed, SEI Technical Report CMU/SEI-2002-TR-020, Software Engineering Institute.
13. Malone, T.W., and Crowston, K. (1994) The interdisciplinary study of coordination, *ACM Computing Surveys*, 26,1, 87-119.
14. Nidumolu, S. (1995) The effect of coordination and uncertainty on software project performance: residual performance risk as an intervening variable, *Information Systems Research*, 6,3, 191-219.
15. Quinn, R.W., and J.E. Dutton (2005) Coordination as energy-in conversation, *Academy of Management Review*, 30,1, 36-57.
16. Schalliol, G. (2001) Challenges for analysts on a large XP project, XP/Agile Universe.
17. Thompson, J.D. (1967) Organization in Action, McGraw-Hill, Chicago.
18. Tiwana, A., and Mclean, E.R. (2005) Expertise integration and creativity in information systems development, *Journal of Management Information Systems*, 22,1, 13–43.
19. Van de Ven, A.H., Delbecq, A.L., and Koenig Jr, R. (1976) Determinants of coordination modes within organizations, *American Sociological Review*, 41,April, 322-338.
20. Weber, M. (1946) Essays in sociology, Oxford University Press, New York.