MANAGING CRITICAL KNOWLEDGE MANAGEMENT ISSUES IN GLOBAL SOFTWARE DEVELOPMENT PROJECTS

Che-Hung Liu, Florida International University, cliu001@fiu.edu
Roman Wong, Barry University, rwong@barry.edu
Yen-Tzu Chen, Nova Southeastern University, yentzu@nova.edu
Hua-Wei Huang, Diwan College of Management, Taiwan. hwawei7@yahoo.com.tw

ABSTRACT

Software development is a complex, knowledge-intensive, and innovative process that frequently takes place in an extremely competitive and highly uncertain environment. For a long time, software developers have used traditional-structured approaches that are more disciplined and focus on coordinated actions to ensure the software goals are met. Recently, this traditional way of software development has faced challenges by an alternative approach, namely, Agile Method. The essences of Agile Method basically include the incorporation of autonomy into software development to nurture the creativity and resourcefulness of developers. In this paper, we identify the critical knowledge management (KM) issues currently arising in many global software development projects and the methods for their management. We also present a framework for the adoption of an agile method to manage complex software development projects in a global environment. This framework can be characterized as 'Discipline at the Global Level, Be Agile at the Local Level.'

Keywords: Software Development, Agile Method, Discipline Approach, Knowledge Management

INTRODUCTION

Software development is a complex, knowledge-intensive, and innovative process that frequently takes place in an extremely competitive and highly uncertain environment. For a long time, software developers have used the more traditional structured-approaches for software development. Such approaches emphasize planning and discipline in each of the development stages. Recently, this traditional way of software development has met challenges from an alternative approach, namely, the Agile Method.

The term Agile Method of software development was coined in the 1990's [1]. Its essences basically include the incorporation of autonomy into software development to nurture creativity and resourcefulness of developers. Researchers have suggested that the complex global environment developers are facing today would warrant the adoption of the agility principles in software development teams more than the values of enforcing discipline in software development [2]. Despite the suggested benefit of being more responsive, agility methods have a problem in managing the vast amount of information and knowledge, which were relevant. In this paper we identify the critical knowledge management (KM) issues currently arising in many global software development projects and the methods for their management. We then present a framework for the adoption of an agile approach to manage complex software development projects in a global environment. This framework can be characterized as "Discipline at the Global Level, Be Agile at the Local Level."

DISCIPLINE APPROACH

Discipline approaches have been used to create well-organized memories, history, and experience and enabling coordinated action to ensure the software goals are met. The Discipline approaches view the software development process like a waterfall from the concept to the end product with strong documentation and traceability mandates that appear across requirements, design, and codes [2]. For example, *General Process Standards* - ISO 12207 and 15504 address the software life cycle and ways to appraise software processes, while ISO 9000 is a quality management standard that includes managing software [3]. *Team Software Process* is a structured framework of forms, guidelines, and procedures for developing software, and supports the development of industrial-strength software through efficient team planning and control [1]. In practice, discipline approaches require management support, organizational infrastructure, and an environment where employees clearly understand the importance of common processes to their personal work and the over-all success of the enterprise [2].

AGILE METHODS

The Agile Method is a rapid prototyping and development experience which illustrates that programming is a craft rather than an industrial process. The Agile Method targets on "chaordic" work, which unifies chaos and order in such a way that the work effort defies traditional management practices by using or rejecting normal, traditional, linear planning and processes [2]. One of the most famous Agile Methods is *eXtreme Programming (XP)*, a fairly rigorous method that initially expects all of the practices defined in the method to be followed. These practices include stories, pair programming, simple design, test first, and continuous integration [4]. Other agile methods include Adaptive Software Development (ASD), Crystal, and Feature Driven Development (FDD). The Agile Method treats changes as an equal partner in the project. Changes are accepted, welcomed, and even capitalized upon. The Agile Method also treats deviations from a plan as new information about the project. Also, the Agile Method strongly suggests and supports face-to-face communication. Communication between team members and between the team and customers is a primary necessity. Furthermore, the Agile Method is an ideal way for developers to survive in a rapidly changing and unpredictable business environment because it is a strategy that works to meet the challenges of profiting from a continually changing global environment while still seeking and achieving high quality, high performance, customer-configured services, and workable processes [3].

Boehm and Turner [2] provide comprehensive perspectives to compare the different characteristics of the Agile Method and the Discipline method. In terms of application, the characteristics include primary goals, size, and environment. Management characteristics include customer relations, planning and control, and project communication, while technical characteristics include approaches to requirements of definition, development, and testing. Personnel characteristics cover customer characteristics, developer characteristics, and organizational culture.

THE GLOBAL SOFTWARE DEVELOPMENT PROJECTS

Software development is viewed as a knowledge-intensive process [1]. Not only should software developers have enough professional knowledge to design the software, but also should be able to understand the operational details of business processes. It is also necessary for software developers to realize the input/output of a business process to create real useful knowledge/software for customers. Understanding customers' requirements for software functions and presentations is also a highly knowledge-intensive process. Software developers need to communicate with customers to learn what they want, and also let them stand near and in view to evaluate the software process. Software development has its own problem-solving codes, rules, and methods. Software developers need to comprehend these methods, and talk with other developers to ensure that they are using the same problem-solving codes and rules.

As Information Communication Technology (ICT) has rapidly developed over the past couple of decades, a new globally dynamic environment has resulted in the software project becoming more knowledge-intensive and complex. The global software projects now face more widely-distributed development which needs integration with different software developers around the globe. The communication and language interactions between different software development organizations, and between organizations and customers also remain one of the main challenges, while global projects now can increase its software project sizes, and advance the problem-solving process to allow a more complicated project. To divide a global project into small parts for different purposes in various software divisions is to plan a milestone schedule and also train software developers to know the same methods and principles. In a global software development environment, coordination is difficult yet more important then ever.

QUALITY VS. EFFICIENCY

The Discipline approach improves software quality and the effectiveness of the software process, but this method takes too much time and cost. The effectiveness of a software project is obviously important, but quality is not something that customers are willing to sacrifice. On the other hand, the agile method offers an alternative strategy to improve the speed of the software development process. How organizations adopt traditional methods and/or agile methods and balance the advantages of these two to achieve the effectiveness and efficiency of any software development project remains still an important issue today.

Nidumolu and Subramani [5] demonstrate how to adapt process and structure approaches to control and manage the software development. Controlling

through process concerns rules of performance and the establishment of management procedures that guide activities. Control has two dimensions: Behavior control processes that regulate activities by clarifying the details of specific behaviors involved in task execution, and outcome control processes that regulate activities by clarifying the details of the outcomes of the task execution. Additionally, control through structure is concerned with the levels of standardization and decentralization.

After testing the modes of control in a sample of 56 software firms, Nidumolu and Subramani [5] concluded that software industry managers should rely on outcome control rather than on methods. In addition, their study suggests that "performance is enhanced by establishing uniform performance criteria across projects (i.e., standardization of performance criteria) while giving each project team the authority to make decisions with respect to their own methods (decentralization of methods)" [5]. We can conclude that a mixed organization structure that is disciplined at the global level and agile at the local level will address the managing global software development process successfully and efficiently.

KNOWLEDGE MANAGEMENT ISSUE IN GLOBAL SOFTWARE DEVELOPMENT PROCESS

Knowledge management (KM) is defined as doing what is needed to get the most out of knowledge resources. It is suggested that an organization with smooth knowledge management processes may be more competitive. The knowledge management process is the creation and transferring between two types of knowledge, which are tacit knowledge and explicit knowledge [6]. Tacit knowledge has a personal quality that makes it hard to formalize and communicate. Explicit knowledge concerns knowledge that is transmittable in formal, and systematic language. Nonaka [6] has identified four modes of knowledge conversion that occur during the knowledge creation process (Table 1): Socialization, from tacit knowledge to tacit knowledge; combination, from explicit knowledge to explicit knowledge; externalization, from tacit knowledge to explicit knowledge; and internalization, from explicit knowledge to tacit knowledge. In the global software development process, for example, socialization always happens when software developers share their personal observations (i.e., cultural and language differences) and work experiences (i.e., coding or modification) for information. Externalization happens later when some of the developers' experiences are recorded via documentation (i.e., a software development guide). Combination happens after developers coordinate with other sections of the software organization and the documentation of existing knowledge (i.e., software development guidelines in different areas). Finally, the internalization process happens when developers share documentations through interaction and a process of trial and error, with different aspects of tacit knowledge (i.e., a software training program).

Table 1. Modes of Knowledge Creation

To / From	Tacit Knowledge	Explicit Knowledge
Tacit Knowledge	Socialization	Externalization
Explicit Knowledge	Internalization	Combination

From a knowledge management perspective, it was learned that the agile method does not focus enough on providing the necessary information system supports for sharing explicit knowledge [3]. However, since the global software development process becomes more complex, software developers cannot rely only on their tacit knowledge to solve problems completely. In the software industry, most knowledge belongs to documentation, including the explicit knowledge of modules, codes, contacts, and guidelines, etc. Without documentation and explicit knowledge support, the agile group may not work efficiently because it is only depending on tacit knowledge. Furthermore, when new employees in agile groups accept a training program, it is better for them to accept explicit knowledge. Tacit knowledge is not considered as great a contribution to knowledge effectiveness as explicit knowledge does [7].

MANAGERIAL IMPLICATIONS

A framework, namely "Disciplined Globally, Agile Locally" (see Figure 1), is suggested to balance the Discipline methods and the agile methods in the global

software development process. In the global software development environment, we assume a single agile group cannot face clients directly due to the distributed environment and software project size. The management office in software organizations plays an important role in communicating with clients. Besides, the management office evaluates the performance, manages the overall projects, and is able to extend communication to provide the level of support and coordination required to meld several successful agile subprojects together into a single successful project.

A software organization has many agile groups. Each group has a special member who is responsible for eliciting and organizing knowledge to and from the knowledge repository and for improving member coordination. The agile groups can be in different countries around the world. After a project is assigned, the agile groups retain local decision-making power and problem-solving authority.

The Knowledge-Mart (K-Mart) plays a very specific role in software organization. Five knowledge repositories (KR) store different kinds of knowledge suited for the global software development process. These are output requirements, client knowledge, business processes, software problems and methods, and agile knowledge. Among the K-mart, knowledge sharing utilities, such as document management systems, groupware, e-mails, databases, and workflow management support the knowledge sharing process. The function of K-Mart is not only to improve the sharing of organizational knowledge, including explicit and tacit knowledge, but also to contribute to intra-organizational communication processes. We also suggest the following software organizational needs:

- Need for building a knowledge repository for storing structured and semi-structured knowledge, best practices, etc.
- Need for establishing a special role in the agile group responsible for eliciting and organizing knowledge to/from the knowledge repository.

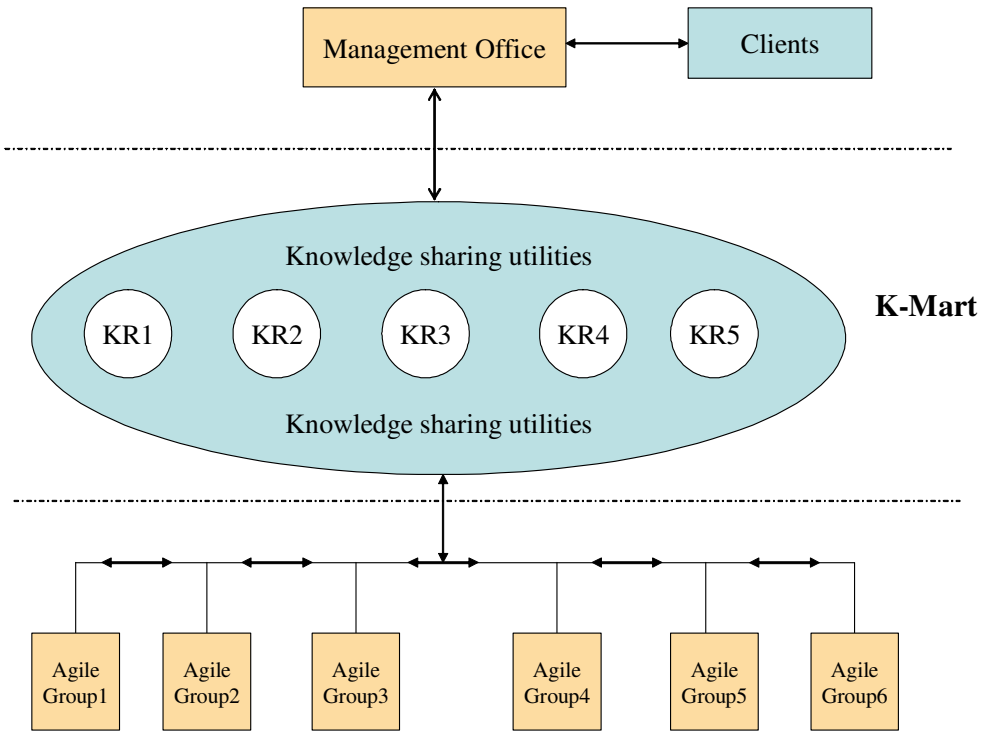

Figure 1. Disciplined Globally, Agile Locally

CONCLUSION

This paper began with the identification and description of two software development methods, namely discipline methods and agile methods. To address the issues related to global software development project, we discussed the importance of balancing efficiency and effectiveness. Depending on the matrix of process and structure control methods, we examined how management should define standard criteria for groups and then give groups more authority to make decisions and solve problems. From this knowledge management process, we determined that agile methods do not focus enough on providing support for sharing and transferring of non-tacit knowledge. Finally we proposed a novel organizational structure called "globally disciplined, locally agile" to balance the agile methods and discipline approaches in the global software development process for great success of software development projects. The inherent limitations of this study should be the current lack of empirical data and the fact that this study excluded some potentially important factors, including leadership and outsourcing, when we analyzed the over-all effectiveness of the global software development process.

REFERENCES

1. Cockburn, A., (2002) *Agile software development*. Boston, MA: Addison-Wesley.
2. Boehm, B. & Turner, R. (2002). *Balancing Agility and Discipline: A guide for the Perplexed*. Boston, MA: Addison Wesley.
3. Koch, A. (2005). *Evaluating Agile software development: Methods for your organization.* Norwood, MA: Artech House, Inc.
4. Boehm, B.& Turner, R. (2003). Using risk to balance Agile and Plan-Driven methods. *IEEE Computer,36*(6), 57-66.
5. Nidumolu, S.R. & Subramani, M.R. (2003). The matrix of control: Combining process and structure approaches to managing software development. *Journal of Management Information Systems, 20*(3), 159-196.
6. Nonaka, I. (1994). A Dynamic Theory of Organizational Knowledge Creation. *Organization Science, 5*(1), 14-37.
7. Becerra-Fernandez, I. & Sabherwal, R. (2001). Organizational Knowledge Management processes: A contingency perspective. *Journal of Management Information Systems, 18*(1), 23-55.