Information & Management 46 (2009) 9–15

Contents lists available at ScienceDirect

Information & Management

journal homepage: www.elsevier.com/locate/im

Virtual organizational learning in open source software development projects

Yoris A. Au *, Darrell Carpenter, Xiaogang Chen, Jan G. Clark

Department of Information Systems and Technology Management, College of Business, University of Texas at San Antonio, One UTSA Circle, San Antonio, TX 78249, USA

ARTICLE INFO

Article history:
Received 3 May 2007
Received in revised form 23 May 2008
Accepted 26 September 2008
Available online 28 November 2008

Keywords:
Virtual organizational learning
Organizational learning curve
Virtual organization
Open source software
Software development
Project performance

ABSTRACT

We studied virtual organizational learning in open source software (OSS) development projects. Specifically, our research focused on learning effects of OSS projects and the factors that affect the learning process. The number and percentage of resolved bugs and bug resolution time of 118 SourceForge.net OSS projects were used to measure the learning effects. Projects were characterized by project type, number and experience of developers, number of bugs, and bug resolution time. Our results provided evidence of virtual organizational learning in OSS development projects and support for several factors as determinants of performance. Team size was a significant predictor, with mid-sized project teams functioning best. Teams of three to seven developers exhibited the highest efficiency over time and teams of eight to 15 produced the lowest mean time for bug resolution. Increasing the percentage of bugs assigned to specific developers or boosting developer participation in other OSS projects also improved performance. Furthermore, project type introduced variability in project team performance.

1. Introduction

Many people can work together on a task regardless of time, geographic location, or organizational affiliation by adopting a virtual approach [14]. Open source software (OSS) development projects exhibit many of the characteristics that make virtual organizations successful, including self-governance, a powerful set of mutually reinforcing motivations, effective work structures and processes, and technology for communication and coordination. Examples of thriving OSS projects include Linux, Apache, and Mozilla. Although seemingly disorganized, and lacking monetary incentives, the development approach is characterized by design simplicity, team work, a visible product, and communication.

But what makes OSS development projects successful? Mockus et al. [13] conducted a case study on the Apache Web server and Mozilla Web browser projects to learn their development process characteristics; they found that projects based on a relatively small core of developers (10–15 people) could be geographically dispersed, yet communicate and function without conflict via a set of implicit coordination mechanisms (i.e. informal email exchange). However, when the number of core developers exceeded this size, other explicit coordination mechanisms (e.g., a code ownership policy) had to be adopted. In a similar study, Huntley [9] used organizational learning to explain the success of OSS projects; he maintained that it decreased time in fixing bugs. However, there were significant debugging differences in Apache and Mozilla, with project maturity as the apparent reason, as opposed to other factors such as project size and number of programmers. Debugging data were modeled to fit a learning curve. Mozilla, an emerging project, was characterized as having improvements due to learning effects present in their team. Both these authors pointed out significant differences between the projects.

Our intent was to extend and refine their work by including a much larger number of OSS development projects of varying size (in terms of the number of developers involved) and type (from simple file management software to complex enterprise software suite). Specifically, we included 118 OSS projects in our sample. By focusing on multiple projects of varying size and type, we were better able to characterize OSS projects. Our study was initiated to answer the following research questions:

(1) Are learning effects universally present in OSS projects?
(2) What are the factors that affect the learning process?

We used the number and percentage of resolved bugs and bug resolution time to measure learning effects. However, we also looked at how different project types, number of developers (project team size) and their experience, and the intensity of

* Corresponding author. Tel.: +1 210 4586337; fax: +1 210 4586305.
E-mail address: yoris.au@utsa.edu (Y.A. Au).

0378-7206/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.im.2008.09.004

assigned bugs affected the learning rates. Data for this study were obtained from the SourceForge.net[1] database.

2. Theoretical framework and hypotheses

We developed several hypotheses based on theories that relate to virtual organizational learning. Our first hypothesis seeks to show that organizational learning exists in OSS development projects. The subsequent hypotheses seek to explain the variation of learning rates observed across projects.

2.1. Organizational learning curves

Group learning curves were first observed in the 1940s during construction of ships and aircraft [22]. The time required to build a complex product decreased at a diminishing rate as more products were produced.

Fiol and Lyles [6] postulated that there are two levels of organizational learning: higher- and lower-level. The first focuses on re-defining the overall organizational strategy under ill-defined context; examples include developing a new organizational culture and re-establishing organizational priorities [3]. Conversely, that lower-level learning focused on specific organizational behaviors and constraints within existing organizational rules, suggesting that minor managerial adjustments, improved problem-solving skills and that the development of formal rules were examples of it. This type of learning is primarily a process of repetition [5].

We consider debugging as a way that organizational experience is accumulated in OSS development teams, thus establishing the software development learning curve. This is lower-level learning where developers repeatedly scan, review, and/or modify program code. As the team gains experience, it exhibits its learning curve by decreasing its average time to resolve a bug. Therefore, we hypothesized that:

H1. As the number of bugs resolved to date increases, the average bug resolution time decreases.

2.2. Cognitive capital and developer's OSS experience

Cognitive capital consists of expertise and the knowledge about how to apply expertise in solving a problem. Over time, people develop it as they learn the skills, knowledge, specialized dialogue, and norms of their work and interact with others who share the same practice [18].

OSS developers can be concurrently involved in more than one project, allowing them a greater opportunity to work with others, learn about the norms of OSS development, and accumulate more experience. Overall, developer's OSS expertise increases with the number of projects in which they are involved. This translates into larger cognitive capital that can be shared with other team members to improve team performance.

Social capital is defined as the number of the ties or interactions that an actor (e.g., a developer) has with another in a social event within a social network or community. For example, Okoli and Oh [15] found a significant relationship between developer performance on Wikipedia and their social ties within the Wikipedia community. Grewal et al. [7] measured social capital as "network embeddedness" using parameters derived from the number of projects in which an OSS developer had been involved. We therefore used *number of OSS projects* as a measure for OSS developer experience.

This lead to the hypothesis:

H2. Teams with more experienced OSS developers resolve bugs faster.

2.3. Task ownership

Task ownership occurs when a task performer takes personal interest and responsibility for it. Its degree can affect how the task is accomplished. It improves team effectiveness and facilitates individual learning; for example, students exhibit a sense of individual accountability when their grade is based on individual efforts in a group project. This also helps to eliminate non-participants.

The relationship between task ownership and individual/team performance can be explained by Goal-Setting Theory, which maintains that task ownership helps task performers clarify their task goals [12]. In turn, these help performers focus attention on goal-related activities, thus improving performance. Rasch and Tosi [16] validated Goal-Setting Theory in software development teams.

We hypothesized that:

H3. There is an inverse relationship between increasing the percentage of bugs assigned to specific developers and average bug resolution time.

2.4. Project category

At the time of our research, Sourceforge.net classified its projects into thirteen categories. These included Clustering, Database, Desktop, Development, Enterprise, Financial, Games, Hardware, Multimedia, Networking, Security, SysAdmin, and VoIP. Projects in different categories typically have different complexity and timeliness, affecting their bug resolution times. We hypothesized:

H4. Different project categories have different average bug resolution times.

2.5. Project team size

Prior work on traditional co-located teams has suggested that the appropriate team depends on the nature of the task; for example, Hwang and Guynes [10] reported that large computer-supported groups generated more decision alternatives but took longer to reach a decision. If a team is too small, it does not effectively share the workload but if it is too large, coordination the overhead is large and social loafing becomes a concern.

The relationship between team size and team performance in OSS communities might exhibit a different pattern than that in a traditional setting, because the communication structure is different; the OSS project team generally consists of two subgroups of developers: core developers and code contributors. Core developers make the critical decisions (e.g., when to release the next version and whether or not to implement a new feature). However, to reach consensus on these decisions, intensive communication among core developers is critical and a small number of core developers per project is therefore recommended.

Code contributors produce the code. They receive well-defined subtasks (i.e. bugs), work on them independently, and, when finished, report back to core developers. As a result, the communication structure follows a star topology. The core developers are the central "hub," and all the contributors connect to and through this hub.

[1] Details were avalaible at http://zerlot.cse.nd.edu/mywiki/ Christley and Madey [4] provided further descriptions of the SourceForge.net data set.