Virtual Organizational Learning in Open Source Software Development Projects

Yoris A. Au, University of Texas at San Antonio, One UTSA Circle, San Antonio, TX 78249, USA; E-mail: yoris.au@utsa.edu
Darrell Carpenter, University of Texas at San Antonio, One UTSA Circle, San Antonio, TX 78249, USA; E-mail: darrell.carpenter@utsa.edu
Xiaogang Chen, University of Texas at San Antonio, One UTSA Circle, San Antonio, TX 78249, USA; E-mail: trent.chen@utsa.edu
Jan G. Clark, University of Texas at San Antonio, One UTSA Circle, San Antonio, TX 78249, USA; E-mail: jan.clark@utsa.edu

1. INTRODUCTION

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 (Markus et al. 2000). Examples of thriving OSS projects include the Linux operating system, Apache Web Server, and the Mozilla Web Browser. Many OSS projects have achieved substantial success despite their seemingly disorganized structure (e.g., no central management) and the lack of monetary incentives. Raymond (2001) described the open source method of development as "a great babbling bazaar of differing agendas and approaches… out of which a stable and coherent system could seemingly emerge only by a succession of miracles." The Bazaar development approach is characterized by design simplicity, teamwork, a visible product, and communication (Wagner 2006).

Researchers have studied OSS development to better define the successful characteristics of this particular form of virtual organization. For example, Mockus et al. (2002) conducted a case study on the Apache Web server and Mozilla Web browser projects. They found that projects based on a relatively small core (10 to 15 people) of geographically dispersed developers could communicate and function without conflict via a set of implicit coordination mechanisms (i.e. informal email exchange). However, other explicit coordination mechanisms (i.e. code ownership policy) were required to maintain communication and reduce conflict when the number of core developers exceeds 10-15 people.

In a related study, Huntley (2003) attempted to explain the success of OSS projects using organizational learning effects. He maintained that learning effects were manifested by the decreased time required for fixing bugs. He noted significant debugging differences in Apache versus Mozilla, with the attributing factor being project maturity, as opposed to other measurable factors such as project size or number of programmers. Huntley modeled debugging data from Apache and Mozilla according to learning curve formulas. As noted, Mozilla, an emerging project, exhibited a steady debugging process, with predictable improvements. The results illustrate that the learning effects are present in the Mozilla team. In their attempt to defining OSS success, Crowston et al. (2003) suggested that the number of developers involved in a project was an important indicator of the success because the project can gain momentum going forward only by attracting enough voluntary developers.

Our research seeks to extend Huntley (2003)'s study by analyzing 118 OSS development projects (as opposed to only two in Huntley's). These projects vary not only in size (in terms of the number of developers involved and lines of code developed) but also in type (from simple file management software to complex enterprise software suites). We draw our data from SourceForge.net's vast database. Specifically, we are interested in answering two main research questions. First, are learning effects universally present in OSS projects? Second, what are the factors that affect the learning process? Similar to Huntley (2003), we use the number of reported bugs and bug resolution time to measure the learning effect. We look at how different project types, number of developers and their experiences, and the intensity of assigned bugs affect the bug resolution time, and whether there is a learning curve effect.

2. EMPIRICAL MODEL

Based on the Power Law learning curve formula (Wright 1936), and motivated by the models in Argote et al. (1990) and Huntley (2003), we developed a log-linear regression model with both qualitative and quantitative variables:

$$\mathrm{h}\ \mathit{MeanResTime}_{it} = \mathrm{a}_0 + \mathrm{a}_1 \mathrm{h}\ \mathit{CumResBugs}_{i} + \mathrm{a}_2 \mathrm{h}\ \mathit{AvgDevExp}_i + \mathrm{a}_3 \mathrm{h}\ \mathit{PctAssignedBugs}_{t}$$
$$+ \sum_{i=1}^{12} \mathrm{b}_i\ \mathit{ProjCat}_i + + \sum_{j=1}^{3} \mathrm{g}_j\ \mathit{ProjSize}_j + \mathrm{e}_i$$

Where:

MeanResTime $_{it}$ = Mean time to resolve the bugs of Project *i* reported in Week *t*
CumResBugs $_{it}$ = Cumulative resolved bugs of Project *i,* including Week *t*
AvgDevExp $_i$ = Average number of other projects each developer in Project *i* has worked on
PctAssignedBugs $_{it}$ = Percentage of assigned bugs in Week *t* of Project *i*
ProjCat $_i$ = Category of Project *i*
ProjSize $_i$ = Size of Project *i*, measured in terms of the number of developers in the project (1 developer; 2-4 developers; 5-10 developers; >10 developers)

Our model tests the following hypotheses:

H_1: As the number of bugs resolved to date increases, the average bug resolution time decreases.
H_2: Increased developer experience decreases average bug resolution time.
H_3: Increasing the percentage of bugs assigned to specific developers decreases average bug resolution time.
H_4: Project type has an effect on average bug resolution time.
H_5: Project size has an effect on average bug resolution time.

3. DATA COLLECTION AND DESCRIPTION

We collected data from SourceForge.net's repository of more than 100,000 projects.[1] SourceForge classifies projects according to the following categories: database, development, desktop, games, hardware, enterprise, financial, games, multimedia, networking, security, system administration, and VOIP. To ensure an appropriate cross-section of Open Source projects were included in our sample, we identified the top 50 projects in each of these categories based on two factors: development status and site rank. The first factor prevented "conceptual" projects with no event reports from reducing the set of usable responses. The second factor produced the best projects based on SourceForge.net's internal ranking system. The ranking system uses three sub-factors 1) traffic, 2) communication, and 3) development to determine an overall rank of projects. The multi-factor ranking system enhanced sample validity by dropping older and less active projects. This produced a sample representative of the current state of Open Source development. Based on these rankings, we collected a "snapshot" of the top 50 projects in each category on March 9, 2006. Note that some projects were cross-listed in multiple categories.

We determined the final dataset by 1) assigning cross-listed projects to their most appropriate category, 2) removing projects with less than two years of data, and 3) removing projects with less than 100 bug reports. This reduced the sample to 118 projects.

Each project has a "bug report", which provides a generic description for project events including number of: 1) bugs, 2) support requests, 3) patches, and 4) feature requests. Each bug also has a status such as open, closed, deleted or pending. An important measure of organizational learning is a comparison of the ratio between reported and closed bugs. After applying all project selection criteria our final pool of bugs included 91,745 reported bugs and 73,253 resolved bugs. We then aggregated the data to produce weekly averages for each project. This resulted in a dataset capturing 16,175 project-weeks of information.

We also collected information about the developers associated with the projects. This included the number of developers for each project, as well as information regarding developers registered for more than one project. We used this data to test our hypotheses related to number of developers and developer experience.

Bottlenecks

Collecting bug data from the SourceForge.Net repository proved to be the greatest project challenge. Data was only accessible through a limited web-based interface. As a result, we had to run multiple small queries and compile results into a single database. We often faced connectivity problems, which hindered our data retrieval efforts. At one point, the database was unavailable for several days because of a system upgrade. Fortunately, the upgrade alleviated some of our data retrieval problems. Once retrieved, the data had to be formatted, subjected to a number of intermediate calculations and aggregated to produce the desired data set. This eventually entailed a process of more than 200 individual steps.

4. PRELIMINARY RESULTS

Preliminary results show support for each of the five hypotheses:

- Average bug resolution time decreases as the cumulative number of bugs resolved increases (H1)
- Average bug resolution time decreases as developer experience increases (H2)
- As percentage of bugs assigned per developer increases, average bug resolution time decreases (H3)
- Project types "SysAdmin" and "Hardware" have the lowest bug resolution times (H4)
- Projects utilizing 2-4 developers have the lowest average resolution time (H5)

Following is a brief overview of some of the major points.

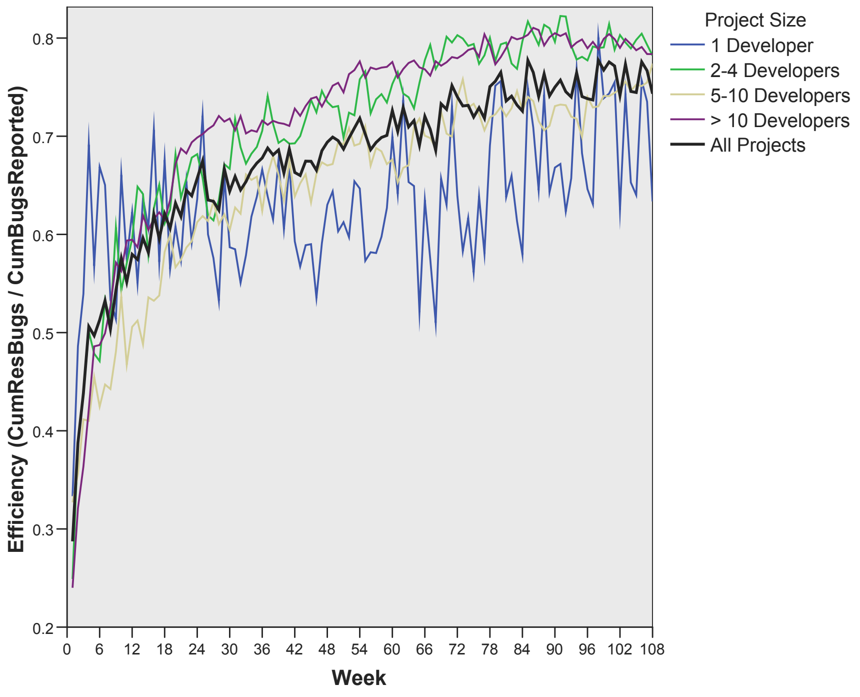
To test the impact of project size, we divided the projects into 4 categories of project size consisting of 1, 2-4, 5-10, and >10 developers, with project size of 1 developer as the reference category. The results indicate that all project sizes have lower resolution times than the reference category, with projects utilizing 2-4 developers having the lowest average resolution time. The average resolution time increased for projects with 5-10 developers and then decreased slightly for those projects with more than 10 developers.

Regression analysis resulted in a negative coefficient ($p < 0.000$) for *CumResBugs*, providing support for H1. Average bug resolution time decreases as the cumulative number of bugs resolved increases. This is in contrast to Huntley's (2003) finding. This finding indicates the presence of a learning curve effect, which is measured by improvements in mean cycle time as more bugs are resolved. A closely related measure is adaptive learning, which is the ratio of cumulative resolved bugs to cumulative reported bugs (Huntley 2003). The graph in Figure 1 provides evidence of an adaptive learning process in the projects, but the process varies based on project size. In particular, projects with a single developer learn faster and thus achieve better efficiency in a shorter period; but over time, they become less efficient relative to projects that employ a group of developers. Projects with 2 to 4 developers demonstrate the best efficiency over time, followed closely by projects with more than 10 developers. It is also interesting to note that the variability of efficiency decreases substantially as the number of developers increases.

5. CONCLUSION

Our preliminary results show there are learning effects in OSS projects. They also show that other factors such as developer experience, project type, project size, and the percentage of bugs assigned to specific developers affect the bug

Figure 1. Comparison of project size efficiency per project week



resolution time and thus the learning curve. Space limitations prohibit further discussion at this time. We will provide a detailed discussion of each hypothesis and its implications at the conference.

REFERENCES

Argote L., Beckman, S. L., & Epple, D. (1990). The persistence and transfer of learning in industrial settings. Management Science, 36(2), 140-154.

Christley, S., & Madey, G. (2007). Analysis of activity in the open source software development community. *Proceedings of the 40th Hawaii International Conference on System Sciences*. Los Alamitos, CA: IEEE Computer Society Press.

Crowston, K., Annabi H., & Howison, J. (2003). Defining open source software project success. In March, S. T., Massey, A. P., & DeGross, J. I. (Eds.) *Proceedings of the 24th International Conference on Information Systems*, Seattle, WA.

Erikson, J. M., & Evaristo, R. (2006). Risk factors in distributed projects. *Proceedings of the 39th Hawaii International Conference on System Sciences*. Los Alamitos, CA: IEEE Computer Society Press.

Huntley, C. L. (2003). Organizational learning in open-source software projects: An analysis of debugging data. IEEE Transactions on Engineering Management, 50(4), 485-493.

Markus M. L., Manville B., & Agres C. E. (2000). What makes a virtual organization work? *Sloan Management Review 42*(1), 13-26.

Mockus A., Fielding R. T., & Herbsleb J. D. (2002). Two case studies of open source software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology, 11*(3), 309-346.

Raymond, E. S. (2001). The cathedral and the bazaar. *First Monday, 3*(3). Retrieved Jan 03, 2007, from http://www.firstmonday.org/issues/issue3_3/raymond/.

Wagner, C. (2006). Breaking the knowledge acquisition bottleneck through conversational knowledge management. *Information Resources Management Journal, 19*(1), 70-83.

Wright, T. P. (1936). Factors affecting the cost of airplanes. *Journal of the Aeronautical.Sciences, 3*, 122–128.

ENDNOTE

1 Details on SourceForge.net's database are avalaible at http://zerlot.cse.nd.edu/mywiki/ ("SourceForge Research Data Archive: A Repository of FLOSS Research Data"). Christley and Madey (2007) provide further descriptions of the SourceForge.net data set and discuss various data mining techniques that can be applied to the data.