Chapter 12

Communication Network Characteristics of Open Source Communities

David Hinds
Nova Southeastern University, USA

Ronald M. Lee
Florida International University, USA

ABSTRACT

Empirical research has shown that social network structure is a critical success factor for various kinds of work groups. The authors extended this research to a new type of work group—the open source software project community—with the objective of exploring the role of communication networks within these intriguing projects. Using archival data from 143 open source project groups, the authors compiled six measures of social network structure and analyzed these in relation to four measures of group success. This study found that the social network structures of these project communities did not appear to be critical success factors at all, but rather they had no significant impact on success or their effect was opposite of that seen in prior studies of work groups. Various conjectures were suggested that might explain these results, offering opportunities for further research.

INTRODUCTION

In the Art History (Kunsthistorische) Museum in Vienna hangs a famous painting by Pieter Breughel (1563), his vision of the Tower of Babel, supposedly built around 2000 B.C. According to the Biblical account, construction on the tower failed when God scrambled up the languages of the workers. They could no longer coordinate. Work groups, then as now, depend on effective communication for their success. This seems to be especially true of software projects (Figure 1).

Back in the 1970’s, Fred Brooks, project manager for what was then the largest software proj-
ect ever, the operating system for the IBM 360, wrote a classic book, *The Mythical Man Month* (1975/1995). In it, he formulated Brook’s Law: “adding manpower to a late software project makes it later”. This has two aspects. One is the learning curve of new staff: it takes time to learn the complexities of the project. The other is the combinatorics of communication. The number of different communication channels increases exponentially with the number of people involved.

A key factor for project management of any kind is the complexity of the artifact being produced. Brooks (1986) claims that large software systems are among the most complex: “Software entities are more complex for their size than perhaps any other human construct because no two parts are alike (at least above the statement level). If they are, we make the two similar parts into a subroutine—open or closed. … The complexity of software is an essential property, not an accidental one” (Brooks, 1986). As one would expect, composite systems are correspondingly more complex: “A programming systems product takes about nine times as much effort as the component programs written separately for private use” (1995, Ch1).

Brooks, and many others after him, have argued that the only way around the software productivity bottleneck is improved design: “My rule of thumb is 1/3 of the schedule for design, 1/6 for coding, 1/4 for component testing, and 1/4 for system testing” (Brooks, 1995).

Recent research, for example Pendharkar and Rodger (2009), indicates that current software work still encounters many of the same communication difficulties as in the 1970’s. However, they note that these may be mitigated by improved methods such as the use of CASE tools. Paulley (2009) notes that these gains may nonetheless be offset by the added communication difficulties of geographically dispersed software development teams.

Against the backdrop of conventional software project management, the notion that software might be developed based on an ad-hoc, evolutionary style framework seems implausible. Eric Raymond (1999b) comments:

*In The Mythical Man-Month, Fred Brooks ... argued that the complexity and communication costs of a project rise with the square of the number of developers, while work done only rises linearly. Brooks’s Law has been widely regarded as a truism. But ... the process of open-source development falsifies the assumptions behind it—and, empirically, if Brooks’s Law were the whole picture Linux would be impossible.*

This is a remarkable observation. Is it really the case that open source projects do not suffer the same communication challenges as traditional software projects? Or is it perhaps that some communication structures work better than others?

In this paper we present a study that examines the communication structures of open source project communities. In particular, we explore the relationship, if any, between the communication network structure of an open source software project community and the success of the project.

Our approach employs techniques of social network analysis. We focused on social network structural factors that past research has shown to most strongly affect work group success and
Related Content

Are Developers Fixing Their Own Bugs?: Tracing Bug-Fixing and Bug-Seeding Committers
www.igi-global.com/article/developers-fixing-their-own-bugs/62098?camid=4v1a

A Perspective on Software Engineering Education with Open Source Software
www.igi-global.com/article/a-perspective-on-software-engineering-education-with-open-source-software/101203?camid=4v1a

Sleight of Hand or Global Problem: The Two Sides of the Net Neutrality Debate
www.igi-global.com/chapter/sleight-of-hand-or-global-problem/130783?camid=4v1a

Rethinking Expertise in the Web 2.0 Era: Lessons Learned from Project Durian
www.igi-global.com/chapter/rethinking-expertise-in-the-web-20-era/121012?camid=4v1a