December 2006

Three Research Essays: Organizational Learning Structure and Motivations within Open Source Software Development

Derek Hillison
Michigan State University

Three Research Essays: Organizational Learning, Structure and Motivations within Open Source Software Development

Derek Hillison
Michigan State University
hillison@msu.edu

ABSTRACT

This work proposes a multi-method, multi-theoretic approach to increasing our understanding of open source software development. Three research projects explore different aspects of the open software development process. A computer-based simulation allows the investigation of consequences from organizing the work of software development in either traditional hierarchical form or in the emerging open-source organizational form. Organizational learning is studied in the open source context through the examination of routines before and after a crisis in software functionality. Role theory emerges as an explanation of both increasing and decreasing motivations for individual developers in the open source community. These essays give insight into micro-behaviors, processes of learning and implications of organizational design on the performance of software development organizations.

Keywords

IS Development Strategies, Open source software, Organizational Structure, Organizational Learning

INTRODUCTION

Three research projects based on theory from micro-organizational behavior, organizational theory, and organizational learning will increase our understanding of the context, work, and organizational structures of software development. This strategy provides a multi-method, multi-theoretic perspective (see Table 1) that will give scholars and practitioners insight into the micro-behaviors, macro-outcomes, and implications of organizational design on the performance of software development organizations. The open source context provides an opportunity to study the core phenomena of information systems development, but its investigation informs scholars and managers involved in other types of organizations as they seek to assimilate participative features common to the open source archetype.

The first research essay examines and compares organizational structures of open source software development with more traditional software development. Focusing on task assignment and priority, a simulation is proposed based on modeling features of the Garbage Can (Cohen, March, & Olsen, 1972) and coordination (Malone, 1987; Malone & Smith, 1988) perspectives. A general contingency theory (Lawrence & Lorsch, 1967) approach compares performance from a variety of measures, and allows discussion of the tradeoff between organizational design features under typical environmental conditions.

The second research essay utilizes secondary source data (SourceForge) to examine organizational learning of open source software organizations. A routines-based view of the organization focuses on the recognizable, repetitive patterns of action that take place within an open source organization. Specifically, enumerating the sequence of bug fixing behaviors allows a comparison of these routines before and after an environmental shock. Changes in these routines as a response to the shock will demonstrate one aspect of organizational learning and adaptation.

The third essay employs role theory and other micro-organizational behavior theories to explain both increasing and decreasing participation of persons involved in open source development. Despite a focus on the phenomena of open source development, implications for traditional software development and other organizations will result from the examination and integration of management theories in this context. As it becomes more difficult for organizations to monitor and reward specific behaviors, interest increases in motivating organizational citizenship and volunteer behaviors.

Essay Title	Research Questions	Theory Base	Methodology
1. Open Source and Traditional Software Development: Garbage Can vs. Hierarchy?	• What are the performance implications of the organizational design tradeoffs between open source and traditional development?	• Organizational Design and Structure	Computer based simulation
2. Open-Source Software Development as a Learning Organization: A Proposed Empirical Investigation of Routines Following Software Failures	• Do open-source organizations have routines? • Do these routines change in response to the environment? • Are changes in routine stable? • What is the performance impact of these changes?	• Organizational Learning • Theory of Routines	Sequence analysis of routines
3. Role-Identity Theory: Why developers increase and decrease participation in open source projects	• What motivates open source developers to increase and decrease their participation?	• Role Theory • Organizational Citizenship Behaviors	Conceptual

Table 1—Proposed research essays

ESSAY 1: ROLE-IDENTITY THEORY: WHY DEVELOPERS INCREASE AND DECREASE PARTICIPATION IN OPEN SOURCE PROJECTS

Many developers of open-source software are unpaid, volunteering their time and expertise on various projects or software programs and subcomponents. Despite this lack of pecuniary reward, these developers write and disseminate useful code. Economic rationality would predict too few unpaid contributions to this public good compared to that observed. Why do people donate their time and expertise to these projects, absent a monetary reward? Why do they continue to make unpaid contributions? Given their participation, what would cause them to reduce the level of their contribution?

There have been many investigations into the motivations of developers of open-source software. Empiric observation has pointed to reputation, gift-culture and other ascribed motivations from developers such as recreation (Hertel, Niedener, & Herrmann, 2003; Hemetsberger, 2001), and use-value (Hemetsberger, 2001; Mittal & Myung-Soo, 1989). Theory-based empiric research found support for helping and volunteer explanations (Il-Horn, Roberts, & Slaughter, 2004) and learning (Lakhani & Von Hippel, 2003).

One answer to the research questions can be found in role identity theory (Turner, 1968; Stryker, 1980; Stryker & Burke, 2000). People seek opportunities to validate their self-perceived identity, through the process of self-verification (Swann, 1987; Stets & Burke, 2000). The process of self-verification is based on the link between self-views and their fit with feedback from other participants in the social realm. People seek to confirm their perceptions of identity from the perceptions and reactions from others (Swann, 1987). Discrepancies between perceived self and related social feedback are associated with negative affect, congruencies with positive affect (Stets & Burke, 2000).

Swann (1987) outlines two general strategies that allow people to achieve self-verification: manipulation of social environments to ensure the survival of their self-views, and distortion of feedback from others. Developers manipulate their social environment by choosing to participate in projects that reinforce their self-view of being a 'programmer' or contributor. This self-verification process helps lead a contributor to situations where their efforts are most valued, and to withdraw from situations where their efforts are chastised or ignored.

The proposed theory will have practical implications for those seeking to improve the participation of developers in open source projects. Other organizations that depend on volunteers or unpaid organizational citizenship behaviors would also likely benefit from extension and integration of role-based and other explanations for the motivations of voluntary participants. The use of management theories to predict outcomes in such a unique context as open source allows a contribution to both the reference discipline and the information systems development literatures.

ESSAY 2: OPEN-SOURCE SOFTWARE DEVELOPMENT AS A LEARNING ORGANIZATION: A PROPOSED EMPIRICAL INVESTIGATION OF ROUTINES FOLLOWING SOFTWARE FAILURES

This research will observe organizational learning by studying the impact of a external change event that prevents the proper operation of the software product under development. Purveyors of instant messaging clients such as MSN, AIM, ICQ, and Yahoo! have formed proprietary networks for communication. Several open source organizations created multi-protocol clients that could communicate over several of these networks at once. At various times, MSN, AOL and Yahoo! have changed the way clients are able to communicate over their networks, temporarily blocking third-party clients from communicating with proprietary clients.

This research proposes to enumerate the patterns of activity surrounding the development of these open source messaging clients and the impact of externally initiated protocol changes on these activities. Since the events occurred several times in the history of the development of the software, the organization can demonstrate learning by a change in the patterns of activity surrounding subsequent function-blocking events, after the first one and an improvement in performance of the organization in dealing with these events.

Many researchers perceive the organization as a bundle of routines (Grant, 1996b; Grant, 1996a; Nelson & Winter, 1982; Pentland & Rueter, 1994). How would the open source context challenge the conceptualization of routine within traditional contexts? Some early evidence point to some recognizable patterns of activity in the open source community (Crowston & Scozzi, 2005), but how routine are they? Routines are typically conceived as stable, unchanging patterns of activity such as 'standard operating procedures' but they may also display considerable variety or variability (Feldman & Pentland, 2003). Measuring the variability and stability of the sequences of activity surrounding open source software development investigates fundamental questions about the concept of routines in the organization.

If routines imbed knowledge (Grant, 1996b), the changing of these routines in response to the environment may indicate a change in the knowledge used by a particular organization. Periods of mostly stable routine behavior would be punctuated by disjoints in the patterns of activity, as participants work to embed new knowledge into their routines. Once a new routine is in place, the organization may ossify around it, and increase temporal stability of the routine. This may be one process by which experiential learning can occur at the organizational level. Experiential learning is often plotted as a function of performance over time, in a 'learning curve' (Argote, 1999). One contribution of this research would be to demonstrate the process by which experience, as it imbeds knowledge in routines, drives changes in performance.

ESSAY 3: OPEN SOURCE AND TRADITIONAL SOFTWARE DEVELOPMENT: GARBAGE CAN VS. HIERARCHY?

Open source and traditional software development at first blush appear to be completely different forms of organizing. One takes place within formal organizations, with profit motives and hierarchy, while the other is typified by a general lack of profit motive, intrinsically motivated workers, and much flatter and diffuse authoritative structures. Despite these major differences, the actual work of software development and maintenance is very similar between the two organizational forms. Bug fixes, for example follow a similar process in traditional development (Crowston, 1997; Pentland, 1992) as that in open source development (Crowston & Scozzi, 2005; Crowston, in press). This essay examines the differences in structure that drive coordination effort and the choice and allocation of tasks in both traditional and open source software development organizations. The research question is: Under what conditions does each form of organizing perform better for the development of software?

Contingency theory suggests that there is no one best way of organizing, but rather better performing forms for specific contexts and environments (Lawrence & Lorsch, 1967). The design of each organizational form provides performance tradeoffs under differing scenarios. To explain when each form of organizing would be better, a contingency approach explores outcomes of these structures, and makes predictions about how each form may perform in various environmental conditions (complexity, stability, heterogeneity).

Computer-based simulation provides the rigor and control of an experiment with the ability to manipulate and control many variables that would be impractical or impossible with other research methods. In fact, agent-based models allow the construction of tests of formal theory that would be infeasible utilizing other research methodologies. Many scholars have examined effects of organizational structure on performance using various simulation techniques (Carley, 1995; Cohen, March, & Olsen, 1972; Cohen, 1982; March & Olsen, 1976; March & Weissinger-Baylon, 1986).

This paper evaluates two theories with resulting models and investigates their applicability for predicting outcomes in software development. Both coordination structure (Malone, 1987) and organizational decision-making perspectives (Cohen et al., 1972) have been used to explain outcomes across a wide array of organization types. A simulation model is proposed which incorporates features of each modeling framework, embedded within the context of software development.

This research seeks to improve our understanding of the consequences of organizational design choices. The open source context is such a new phenomenon that existing organization theory has not fully benefited from its investigation. This research also informs information systems development literature as it compares traditional forms of development with emerging ones. Practitioners within both forms of organizing can benefit from discussions of organizational structure consequences. Managers of firms involved in the production of other products could introduce salient features adapted from open source organizations as they seek to capture the benefits of 'open sourcing' their own organizations.

CONCLUSION

The proposed work represents unique and novel contributions to the scholarly discourse surrounding the work of software development. Each project seeks to clarify and understand a different aspect of open source software development, from different theoretic perspectives and methodologies. This strategy builds a more complete picture of software development than one particular study and methodology could provide. Information systems literature is informed through the examination of its core phenomena: software development in the open source context. Organizational theory, behavior and learning add to our understanding of the phenomena of open source, but their use in this context also contributes to these reference disciplines.

REFERENCES

1. Argote, L. (1999). Organizational Learning: Creating, Retaining and Transferring Knowledge. Norwell, Massachusetts: Kluwer Academic Publishers Group.
2. Cohen, M., March, J. G., & Olsen, J. (1972). A Garbage Can Model of Organizational Choice. Administrative Science Quarterly, 17(1), 1-25.
3. Crowston, K., & Scozzi. Coordination Practices Within FLOSS Development Teams: The Bug Fixing Process. Proceedings of the First International Workshop on Computer Supported Activity Coordination .
4. Feldman, M., & Pentland, B. (2003). Reconceptualizing Organizational Routines as a Source of Flexibility and Change. Administrative Science Quarterly, 48, 94-118.
5. Grant, R. M. (1996a). Prospering in Dynamically-Competitive Environments: Organizational Capability as Knowledge Integration. Organization Science, 7(4), 375-387.
6. Grant, R. M. (1996b). Toward a Knowledge-Based Theory of the Firm. Strategic Management Journal, 17, 109-122.
7. Hemetsberger, A. (2001). When Consumers Produce on the Internet: The Relationship between Cognitive-affective, Socially-based, and Behavioral Involvement of Prosumers. Fourth International Research Seminar on Marketing Communications and Consumer Behavior .
8. Hertel, G., Niedener, S., & Herrmann, S. (2003). Motivation of software developers in Open Source projects: an Internet-based survey of contriutors to the Linux kernal. Research Policy, 32(7), 1159-1177.
9. Il-Horn, H., Roberts, J., & Slaughter, S. A. (2004). Why Developers Participate in Open Source Software Projects: An Empirical Investigation. International Conference on Information Systems .
10. Lakhani, K., & Von Hippel, E. (2003). How open source software works: "free" user-to-user assistance. Research Policy, 32, 923-943.
11. Lawrence, P. R., & Lorsch, J. W. (1967). Differentiation and integration in complex organizations. Administrative Science Quarterly, 16, 216-229.
12. Malone, T. (1987). Modeling Coordination in Organizations and Markets. Management Science, 33(10), 1317-1332.
13. Malone, T., & Smith, S. (1988). Modeling the performance of organizational structures. Operations Research, 36(3).
14. Mittal, B., & Myung-Soo, L. (1989). A Causal Model of Consumer Involvement. Journal of Economic Psychology, 10, 363-390.
15. Nelson, R., & Winter, S. (1982). An Evolutionary Theory of Economic Change. Cambridge, Massachusetts: The Belknam Press of Harvard University Press.
16. Pentland, B., & Rueter, H. H. (1994). Organizational Routines as Grammars of Action. Administrative Science Quarterly, 39, 484-510.

17. Stets, J. E., & Burke, P. J. (2000). Identity Theory and Social Identity Theory. Social Psychology Quarterly, 63(3), 224-237.

18. Stryker, S. (1980). Symbolic interactionism: A social structural version. S. Stryker Symbolic interactionism: A social structural version . Menlo Park, CA: Benjamin/Cummings.

19. Stryker, S., & Burke, P. (2000). The Past, Present, and Future of an Identity Theory. Social Psychology Quarterly, 63(4), 284-297.

20. Swann, W. (1987). Identity Negotiation: Where two roads meet. Journal of Personality and Social Psychology, 53, 1038-1051.

21. Turner, R. (1968). Social Roles: Sociological Aspects. International Encyclopedia of the Social Sciences . New York: MacMillian.