3-1-2006

Design for Interpreting the Innovation Occuring Within a Free/Open Source Software Development Community

Lars P. Linden
llinden@bus.ucf.edu

DESIGN FOR INTERPRETING THE INNOVATION OCCURRING WITHIN A FREE/OPEN SOURCE SOFTWARE DEVELOPMENT COMMUNITY

Lars P. Linden
University of Central Florida
llinden@bus.ucf.edu

Abstract

Organizations that sell products and services related to free/open source software (FOSS) have an interest in the development community that produces that software. This paper proposes a design theory for an information system that tracks the innovation of a FOSS development community. The design theory is constructed using Information System Design Theory (Walls, Widmeyer, & El Sawy, 1992) and is based upon three kernel theories: one that describes six design principles that support distributed cognition (Boland, Tenkasi, & Te'eni, 1994), one that provides factors found to contribute to successful innovation (Brown & Eisenhardt, 1997), and one that provides a framework for analyzing the FOSS development approach (Feller & Fitzgerald, 2000). A system based upon this design is argued to be beneficial for an organization that places a high value on information about a given FOSS development community.

Keywords: free software, open source software; open innovation, design science

Introduction

Some organizations participating in a free/open source software (FOSS) development community also produce products and services based upon the community's software. This paper introduces a design theory for an information system to be used by these organizations. The goal of the design is to increase an organization's understanding of the innovation that is occurring within a given FOSS development community for the benefit of the innovation that is occurring within the organization.

This paper describes the environment of such a system and then, following Information System Design Theory (Walls et al., 1992), argues that the design be based upon three kernel theories. A theory of designing a distributed cognition system written by Boland, Tenkasi and Te'eni (1994) provides principles for a system to be used by groups within an organization. A theory describing several factors that contribute to successful innovation, as argued by Brown and Eisenhardt (1997), provides a lens through which the members of the organization can more easily identify the signs of innovation occurring within a FOSS development community. Finally, a framework for analyzing the FOSS development approach, constructed by Feller and Fitzgerald (2000), provides a structure upon which to pattern the development process.

System Environment

The information system is being designed to aid organizations that create value by combining FOSS with new products or services. An example of an organization with this business model is a device manufacturer that increases the performance of the hardware by using FOSS. Another example is a consulting company that increases the utility of a client's system by expertly maintaining the client's FOSS software. These organizations have a keen interest in the development community that produces the FOSS. The proposed design would help such an organization span the organization's boundaries.

It is assumed that the organization following this strategy is currently active in a given FOSS development community, but that a casual analysis of the events taking place within the community is not sufficient for decision-making. It is also assumed that the proposed system is used in conjunction with a system that gathers data by the use of software agents which harvest data from the public servers of the FOSS development community. The challenge, if the proposed system is to be justified, is to find a design that takes the harvested information in all its variety and cost-effectively creates a useful abstraction of the FOSS development community. The environment of the system is depicted in Figure 1 and the elements of the environment are described below to provide a context for the discussion of the design theory.

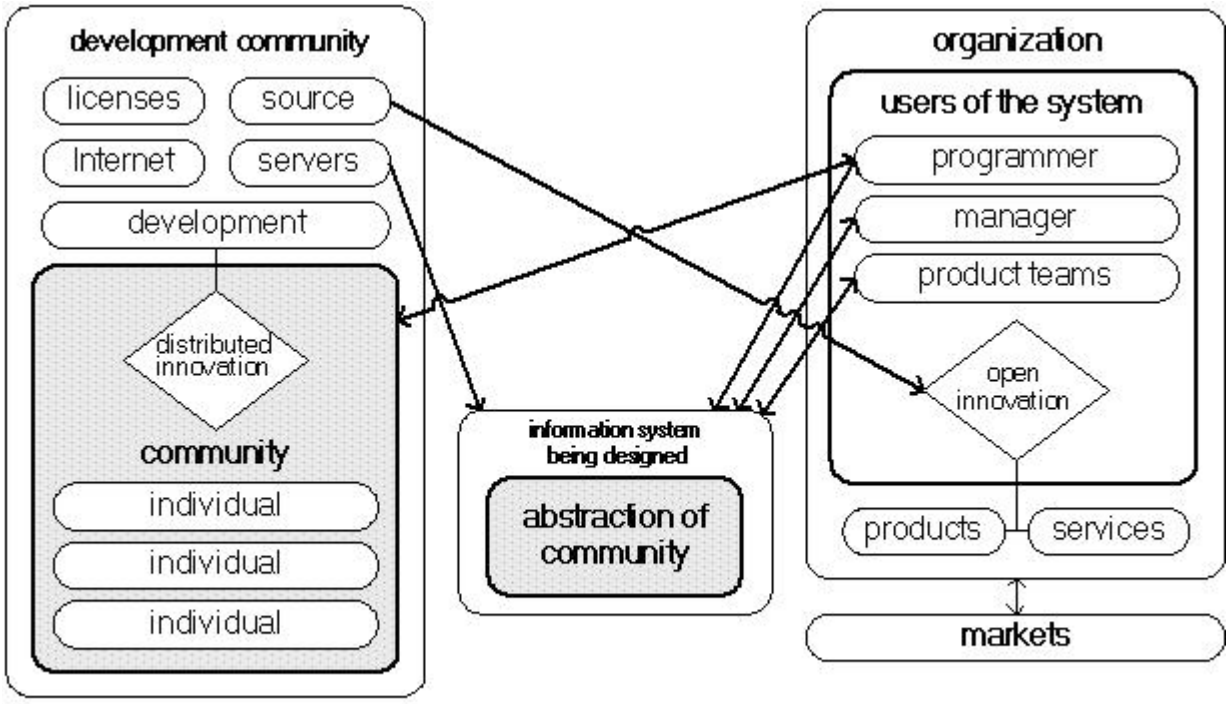

Figure 1. System Environment

Licenses and Source

A FOSS development community has two dimensions, the license which governs the property rights of the software and the use of the Internet which allows a dispersed community to develop the software in a distributed fashion (Kogut & Metiu, 2001). The first dimension, the license, helps define the software being developed. FOSS is a type of computer program that is licensed to protect a range of freedoms. Two widely-discussed definitions of these freedoms are published by the Free Software Foundation and the Open Source Initiative.

The Free Software Foundation provides a definition which states that a program is "free software" when users have a set of freedoms, a set that includes the freedom to study and adapt the source code and the freedom to release improvements to the public (Free Software Foundation, 2004). The Open Source Initiative provides a definition of "open source software" that lists ten criteria, including access to the source code and the allowance to create derived works (Perens, 2005). These are just two of the many FOSS definitions that exist, but they help describe this class of software licenses.

Internet and Servers

The Internet is the second dimension to a FOSS development community (Kogut & Metiu, 2001). One way to describe the extensive use of the Internet by a development community is to point to the many systems that are used. The standard infrastructure of a FOSS project includes a public code archive, project documentation, a bug database,

open mailing lists, newsgroups, and a project Web site (Goldman & Gabriel, 2005). Using these systems and the end-to-end architecture of the Internet, a FOSS development community performs distributed innovation.

Community and Distributed Innovation

"An innovation is an idea, practice, or object that is perceived as new by an individual or other unit of adoption." (Rogers, 1995, p. 11) By providing a way to efficiently identify and track what is new in the development community, the proposed system could provide advantageous information to product and service development. However, identifying what is new may be a challenge.

Weber (2004) discusses a characteristic of distributed innovation that helps highlight the challenge of tracking distributed innovation. Following Bruce Kogut and Anca Turcana's assessment that an end-to-end architecture which enables global software development is not only geographically dispersed but also functionally disperse, Weber points out that distributed innovation is more than just a division of labor, but a situation where the organization of labor is changed. The distributed effort, to a degree at least, has no central command structure. The innovation within a FOSS development community occurs at the edges of the network (Weber, 2004) and the implication for the system being designed is that the structure of the FOSS development community is dynamic.

Open Innovation and the Users of the System

Having described the basic elements of the system environment that are in a FOSS development community, the elements in the organization are now described. First, there is another type of innovation, one labeled "open innovation." Here "open" refers to the boundaries of the organization. When pursuing a strategy of open innovation an organization extends its innovation processes to include parties outside the organization. Open innovation is made possible when an organization opens its boundaries and allows novel ideas to cross in and out of the organization (Chesbrough, 2003; Horwitch, Parikh, & Nina, 2000). Open innovation exists when programmers of the organization submit code to the development community and when product teams use source code from the community as an input to product and service development.

These programmers and product teams, along with their managers, are the users of the system being designed. These people are engaged with a FOSS development community and could benefit from actionable information about an entity that is outside the boundaries of the organization.

Products, Services, and Markets

An organization that abides by the software license and honors any community rules, such as giving and not simply taking, can take advantage of the value created by a development community (Goldman & Gabriel, 2005) and actively pursue an open innovation strategy using FOSS. This leads to the final element of the system environment, value-added products and services finding a market.

Three Kernel Theories

Having described the environment of the system, the first steps toward a complete design theory are taken here by arguing for the use of three particular kernel theories. The approach of creating a theoretical base for a design theory follows that of Walls et al. (1992). Once formulated the design theory is used to create a proof-of-concept which is then used to assess the validity of the design theory (Gregg, Kulkarni, & El Sawy, 2001; Walls et al., 1992). The creation of an artifact to solve an organizational problem and the evaluation of the designed artifact are activities of Design Science research (Hevner, March, Park, & Ram, 2004). Grounding the design in theory is a step toward performing these activities with rigor.

The first kernel theory consists of design principles to support distributed cognition (Boland et al., 1994). The second kernel theory consists of factors argued to contribute to successful innovation (Brown & Eisenhardt, 1997). The third kernel theory is a framework for analyzing the FOSS development approach (Feller & Fitzgerald, 2000). Each of these is now discussed in turn.

Six Design Principles to Support Distributed Cognition

The theory of designing information technology to support distributed cognition proposes using six design principles: ownership, easy travel, multiplicity, indeterminacy, emergence, and mixed forms cognition (Boland et al., 1994). This theory provides an important theoretical foundation for the design because the system is to be used by people who have a variety of roles within the organization.

Also, this theory is appropriate for the system being designed because distributed cognition is a similar feature in both the open innovation initiated by the organization and the distributed innovation occurring within a FOSS development community. Open innovation involves an organization attempting to increase its awareness of the ideas emanating from the experts residing outside the organization, hoping to learn from them. Given this system environment, where there is a need to learn from others, the system would benefit by being based upon the design principles that support distributed cognition.

Factors that Contribute to Successful Innovation

The next kernel theory is argued to be beneficial to the design because it provides a set of characteristics that define innovation. These characteristics are insights resulting from grounded theory building conducted by Brown and Eisenhardt (1997). This kernel theory describes structures and processes that were identified as having contributed to successful innovation. This kernel theory is argued to be appropriate because it considers successful organizations as continuously changing and self adaptive, two traits exhibited by FOSS development communities.

The factors argued to contribute to successful innovation include (1) semi-structures and communications, (2) propositions and probes, and (3) road maps and sequenced transitions. Semi-structures exist where priorities and responsibilities are balanced with undefined improvisational problem-solving processes. These semi-structures are kept in balance by plenty of communications. Propositions and low-cost experiments are used to meet unexpected demands by creating links between the present and the future. Finally, road maps and sequenced transitions are techniques used to actively coordinate the transitions between project iterations. All of these factors contribute to the design by providing ways to focus the users' attention when they attempt to identify innovation.

Framework for Analyzing the FOSS Development Paradigm

The third kernel theory originates from the open source software literature. Feller and Fitzgerald (2000) synthesized a framework for analyzing the open source software development approach. This framework is a combination of Zachman's (1987) Framework of IS Architecture and Checkland's (1989) Soft System Methodology. Zachman's Framework is an IS architecture that draws from Kipling's spectrum of questions and starts by asking what, how, and where. Checkland's CATWOE technique argues for a Soft System Methodology based on a learning organization. Feller and Fitzgerald's framework is useful to the design theory because it provides a theoretical foundation for the development process.

The framework, although not a development methodology, does contain a comprehensive checklist for a development environment and is partially based upon a development methodology. The important aspect of the framework, and why it is argued to benefit the design theory, is that it has a FOSS perspective. By grounding the development process in a framework that has a FOSS perspective, the design would be aligned with FOSS values. By developing with FOSS values, the development would take place in public view. Any criticisms and suspicions from a FOSS community could be discussed and resolved.

Conclusion

This paper introduced a design theory for an information system that aids an organization in interpreting the innovation occurring in a free/open source software development community. The system environment was described as spanning the organization's boundaries and including two processes of innovation. Three kernel theories were argued to provide a theoretical foundation for the design. The next step would be to elaborate the design product, the design process, and the hypotheses that are needed to test the design theory. Following the Design Science methodology, a proof-of-concept would be developed and used to determine if a system based upon this design theory is feasible.

References

Boland, R. J. J., Tenkasi, R. V., & Te'eni, D. (1994). Designing information technology to support distributed cognition. *Organization Science, 5*(3), 456-475.

Brown, S. L., & Eisenhardt, K. M. (1997). The art of continuous change: Linking complexity theory and time-paced evolution in relentlessly shifting organizations. *Administrative Science Quarterly, 42*(1), 1-34.

Checkland, P. (1989). Soft systems methodology. In J. Rosenhead (Ed.), *Rational analysis for a problematic world: Problem structuring methods for complexity, uncertainty and conflict*. West Sussex, England: Wiley.

Chesbrough, H. W. (2003). *Open innovation: The new imperative for creating and profiting from technology*. Boston: Harvard Business School Press.

Feller, J., & Fitzgerald, B. (2000). A framework analysis of the open source software development paradigm. *Proceedings of the twenty first international conference on information systems, Brisbane, December 2000*, 58-69.

Foundation, F. S. (2004). *The free software definition*. Retrieved October 18, 2005, from http://www.gnu.org/philosophy/free-sw.html

Goldman, R., & Gabriel, R. P. (2005). *Innovation happens elsewhere: Open source as business strategy*. Boston: Morgan Kaufmann.

Gregg, D., Kulkarni, U., & El Sawy, O. A. (2001). Understanding the philosophical underpinnings of software engineering research in information systems. *Special Issue of Information Systems Frontiers, 3*(2), 169-183.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MISQ Quarterly, 28*(1), 75-105.

Horwitch, M., Parikh, M., & Nina, Z. (2000). *Open innovation: Transferring lessons from software for modern value creation.* Paper presented at the CISEP Workshop on Innovation and Diffusion in the Economy, Lisbon, Portugal, January 2000.

Kogut, B., & Metiu, A. (2001). Open-source software development and distributed innovation. *Oxford Review of Economic Policy, 17*(2), 248-264.

Perens, B. (2005). *The open source definition*. Retrieved November 10, 2005, from http://www.opensource.org/docs/definition_plain.php

Rogers, E. M. (1995). *Diffusion of innovations*. New York: Free Press.

Walls, J. G., Widmeyer, G. R., & El Sawy, O. A. (1992). Building an information system design theory for vigilant EIS. *Information Systems Research, 3*(1), 36-59.

Weber, S. (2004). *The success of open source*. Cambridge, MA: Harvard University Press.

Zachman, J. A. (1987). A framework for information system architecture. *IBM Systems Journal, 26*(3), 276-292.