Communications of the Association for Information Systems

Volume 17 Article 12

February 2006

Understanding Open Source Software: A Research Classification Framework

Matthew Nelson
Illinois State University, mlnelso@ilstu.edu

Ravi Sen
Texas A&M University, rsen@mays.tamu.edu

Chandrasekar Subramaniam
University of North Carolina Charlotte

Communications of the Association for Information Systems

UNDERSTANDING OPEN SOURCE SOFTWARE: A RESEARCH CLASSIFICATION FRAMEWORK

Matthew L. Nelson
Illinois State University
mlnelso@ilstu.edu

Ravi Sen
Texas A&M University

Chandrasekar Subramaniam
University of North Carolina – Charlotte

ABSTRACT

The success of open source applications such as Apache, Linux, and Sendmail spurred interest in this form of software, its development process, and its implication for the software industry. This interest is evident in the existing research being done to address various issues relevant to open source software and open source methodology. This paper proposes a research classification framework that:

- informs about the current state of open source software research,
- provides a formal structure to classify this research, and
- identifies future research opportunities.

KEYWORDS: open source project, software engineering, software development, open source, economics of open source.

I. INTRODUCTION

Open source software (OSS) allows users to access source code, to freely use the software as they see fit, to improve it, to fix the bugs, to augment its functionality, and to redistribute it. Distribution to other users is typically for free, or at a charge under an Open Source Initiative (OSI) approved license type . These users can, in turn, modify and/or use the software according to their own needs. This approach is in contrast to proprietary and commercial software controlled by the owner (developer or corporation). Some popular examples of open source applications and their proprietary competitors are given in Table 1. The definitions of different OSS license types and the classification of select OSS licenses are given in Appendix A.

Table 1. Examples of Open Source and Proprietary Applications

Application	Proprietary	Open Source
Web Browser	MS Internet Explorer, Netscape	Mozilla (based on Netscape)
Web Server	Microsoft IIS, Netscape	Apache
Application Server	WebMethods (includes JBoss application)	JBoss, Apache Tomcat, Enhydra
Office Suite	MS Office, Corel WordPerfect Office	OpenOffice.org (based on Sun's proprietary StarOffice)
E-mail/ collaboration	MS Exchange, Lotus Notes	Ximian Evolution
Database	Oracle9i, IBM DB2, MS SQL	MySQL, PostgreSQL (both lack the power and depth of industrial-strength Oracle and DB2)

The success of open source projects has led to an increasing interest in understanding this form of software and the way it is developed. The nature of open source software, its concepts, benefits, and challenges are discussed in a recent CAIS tutorial [AlMarzouq et. al., 2005]. Readers new to the OSS area will find this article a useful starting point. Unfortunately, a comprehensive framework to classify existing OSS literature and research issues is lacking in studies on OSS. In particular, a classification framework should help position OSS-related research projects and assist in examining the progress achieved. Furthermore, classifying the existing literature will aid in information retrieval by providing a browsing structure for OSS sub-domains. The analysis presented in this article will help fill these gaps and identify emerging research areas.

II. A CLASSIFICATION FRAMEWORK

Open source software research can be categorized into the following major phases:

- Initiation Phase,
- Ongoing Project Phase, and
- Adoption / Deployment Phase.

In addition, the literature examines coordination mechanisms, participation alternatives, and the impact of OSS related initiatives. Figure 1 is an overview of the OSS phases used in our classification framework.

INITIATION PHASE

Research on the initiation phase addresses questions such as:

- Why are open source projects started in the first place?
- What incentives regulate the initiation phase?

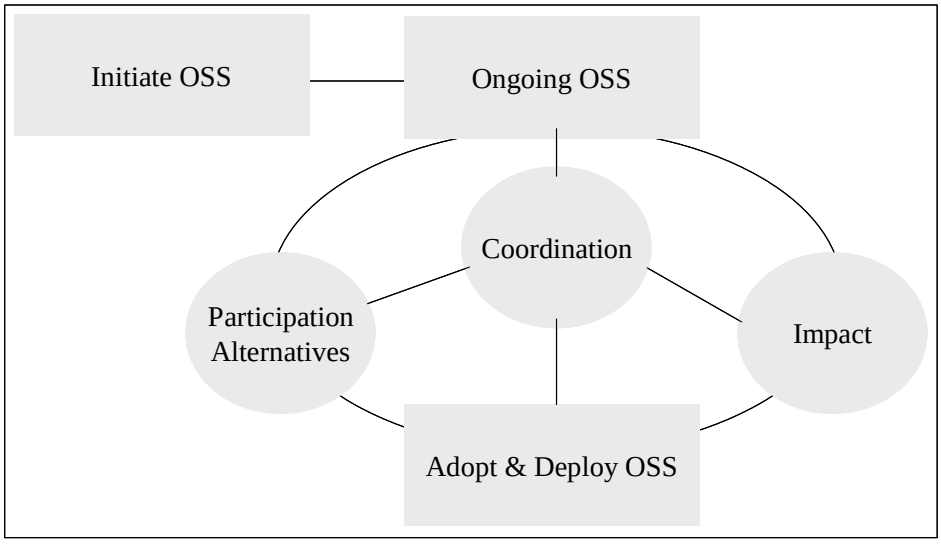

Figure 1. Open Source Software Phases

Research issues include the identification, explanation, and verification of factors that influence the decision to initiate an open source project. These factors include:

- personal/psychological (e.g., reputation enhancement among peers) incentives,
- technical (e.g., need for certain functionality) incentives,
- economic/strategic (e.g., competition, low development cost) rationale, and
- social/political (e.g., desire to improve social welfare) reasons.

ONGOING PROJECT PHASE

Research on the ongoing project phase addresses the continuity of open source projects. For example:

- What motivates individuals and organizations to participate in an ongoing open source project?
- How can individuals and organizations participate? What are the roles they play and the quality of contributions they make?
- Which coordinating and communication mechanisms aid or hinder open source projects?

Currently, the roles and responsibilities of project participants are not clearly defined. In open source projects, the users are also co-developers. Roles are also blurred by early releases, frequent code integration, increased versioning, greater modularization, and dynamic decision making structures [Robles 2004; Raymond 2001].

Coordination is crucial for the transformation of an initiative into an economically viable software application. The spontaneous, decentralized functioning of the open source community represents a challenge to many current notions of coordination. The open source philosophy

assures a “self-correcting spontaneous” organization of work that is “more elaborate and efficient than any amount of central planning could have achieved” [Raymond, 2001]. For example, it is possible to align the incentives of several different individuals without resorting to property rights and contracts. It is also possible to specify everyone’s tasks without resorting to a rigid and formal hierarchical organization.

ADOPTION / DEPLOYMENT PHASE

Studies in the adoption/deployment phase include the examination of strategic, economic, and/or social factors leading towards the diffusion of open source applications. In addition, this phase considers the impact of open source on software quality, productivity, and evolution.

Our proposed research classification framework is based on the three phases of open source software and shows the various research issues in OSS (Figure 2).

The rest of our paper is organized as follows: Section III classifies the research questions on motivations for starting an OSS project. Section IV examines the research on motivations of individuals and organizations to participate in ongoing OSS projects. Section V looks at the technology adoption and diffusion questions in OSS projects. Section VI presents the coordination mechanisms utilized in OSS projects. Section VII considers the various impacts of OSS, including project performance measurement and the impact on the software industrial organization. After identifying research gaps and extensions of the OSS research classification framework, we conclude the paper with future opportunities in this emerging research frontier.

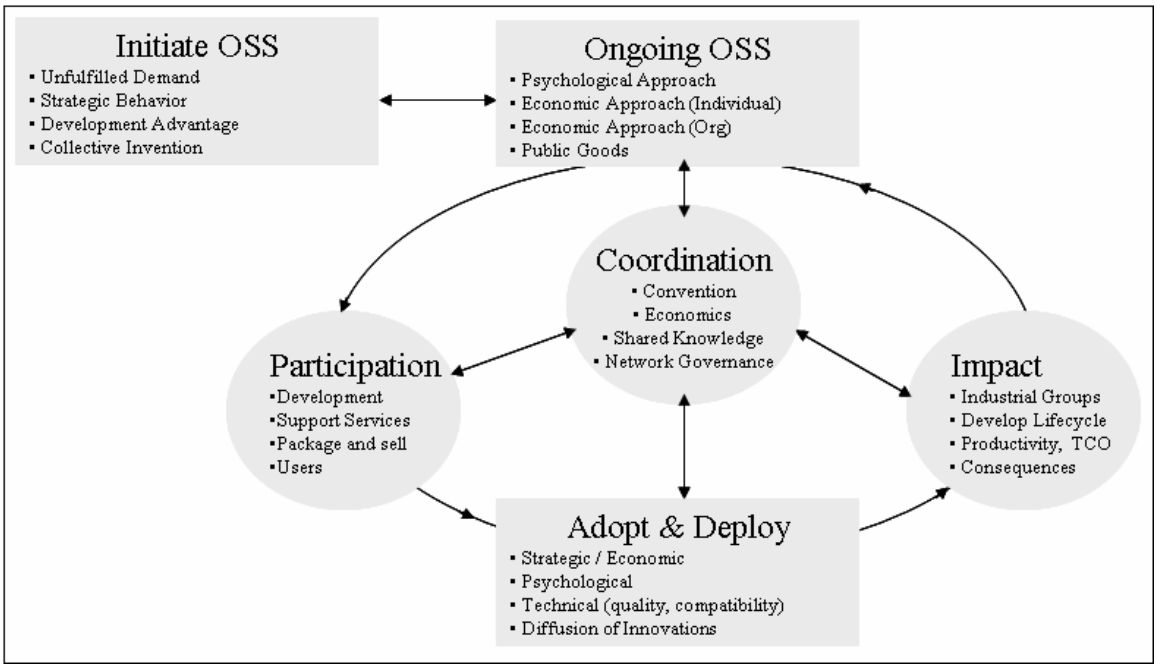

Figure 2. Research Classification Framework based on OSS Phases

III. OSS PROJECT INITIATION PHASE: MOTIVATIONS TO INITIATE AN OPEN SOURCE PROJECT

The existing literature identifies factors that motivate an individual or an organization to initiate an open source software development project. These factors are grouped under the categories of:

- unfulfilled demand,
- strategic behavior,
- development advantage, and
- collective invention.

UNFULFILLED DEMAND

Many open source products are initiated to fulfill demand that cannot be met by proprietary software applications. Open source applications are used to fix problems encountered by IS professionals in their jobs. For example, the Apache webpage server was created because no commercially available software packages served web pages. Similarly, Linus Torvalds initially developed Linux as an inexpensive Unix-like operating system for his 80386-based computer because operating system source code was not available.

STRATEGIC BEHAVIOR

Open source projects are also initiated by businesses to:

- commoditize software that supports their core product, and
- gain first mover advantage among developers who participate in open source projects.

Organizations can increase demand and possibly charge premium prices for their core product by commoditizing the software. For example, IBM commoditized enterprise software with open source initiatives to complement its IT consulting business.

The first mover advantage for open source projects comes from:

- attracting the best people to work on innovative projects
- early development of an open source application, and
- decreased likelihood of a customer switching.

The combination of first mover advantage and developing complementary software for its own core product and services can preempt competition that relies solely on proprietary software.

DEVELOPMENT ADVANTAGE

Individuals and organizations initiate open source projects because they believe they can lower the cost of developing, debugging, and enhancing a software application, and provide users with a better quality application. For many open source projects, a virtual community of developers grows around the software. The project initiator then incurs lower overhead because of unpaid, outsourced work and is closer to customers who actually use the products. Organizations that initiate an open source project (but do not become involved with the application development process) still encourage the diffusion of these applications among potential users. For example, hardware manufacturers may choose to open source unprofitable software accessories (such as hardware drivers). Many Linux hardware drivers were developed independent of their hardware vendors.

Alternatively, organizations may choose the open source route because they believe it results in better quality software. Since many programmers are also users of the product, they have a vested interest in making the product the best it can be. These software developers, also referred to as use-developers, are *capable of* and *interested in* advancing the product. They contribute code, make comments, and participate in discussions. Furthermore, there is the adage based on Linus's Law of Software Engineering, i.e., "given enough eyeballs, all bugs are shallow*."* [Raymond 2001]. The more people look at a piece of code, the more likely one of them will find a bug before it becomes a major problem.

COLLECTIVE INVENTION

Viewed historically, open source software is a type of innovation process called "collective innovation" [Allen, 1983]. In "collective innovation," rivals freely release pertinent information about how they solved non-trivial technical problems [Nuvolari, 2003]. Each rival uses the received information to improve the basic common technological layout. Allen [1983] argues that a pattern of collective invention requires three essential factors:

- Users can choose between a leading edge but riskier design or an older but safer design.
- A mechanism based on reputation, assuring the existence of mutual individual incentives to disclose technical information to outsiders.
- The disclosure of information should not prevent the owners of the technology or the innovation from reaping economic benefits from their innovations.

All these conditions are met by OSS. For example, Linux users can either use even-numbered versions which are rigorously tested and stable, or use odd-numbered versions with more features, but not certified as fully stable. Every OSS project maintains documentation that illustrates the contribution from developers involved with the project. The initiators of OSS projects reap economic benefits from their innovations. Thus, one can argue that OSS projects are started as collective invention, which results in applications that benefit not only the contributors but also the industry that adopts OSS.

IV. OSS PROJECT ONGOING PHASE: MOTIVATIONS TO PARTICIPATE IN AN OPEN SOURCE PROJECT

Participating in an OSS project can manifest in several ways, including joining the project as a developer, providing financial support, promoting the project, and incorporating OSS platforms in internal systems. Current literature takes the following approaches to address the motivational factors that influence an individual and/or an organization to join an ongoing open source project.

PSYCHOLOGICAL APPROACH

The psychological approach is used to explain an individual's motivation to join and is largely based on the assumption that developers are motivated by the recognition of their peers and not necessarily by profit-motives [Bonaccorsi and Rossi, 2002]. Software programming is considered a creative problem-solving activity, especially among programmers [Raymond, 2001], who believe that "*the value of a programming contribution can be "properly" appreciated only by other programming experts."* Open source projects provide programmers an opportunity to "display" their skills and to be acknowledged and appreciated by a large audience. For instance, Linux programmers have evolved to be highly disciplined, ethical and competitive, with a pecking order that excludes all but the best contributors [Koch, 2003]. This desire for peer recognition represents one of the most important individual incentives for joining open source projects [Raymond, 2001]. Such explanations can also find theoretical support in social psychology [Clary et al., 1998; Mauss, 1967] including social exchange theory [Blau, 1964], psychological contract theory [Rousseau, 1995], helping behavior and volunteerism [Benson et al., 1980; Clary and

Orenstein, 1991], and work design [Hackman and Oldham, 1980]. Both social exchange theory and psychological contracts theory are premised on the notion of an exchange relationship between two actors where the relationship is based on norms of reciprocity. Researchers should be cautious when applying these theories in the OSS context because empirical studies on open source software projects have found little evidence of reciprocity [Lakhami and von Hippel, 2003]. Another approach to explain the role played by peer-recognition in motivating individuals' participation is from academic and scientific research. The reward system based on peer reviews in OSS is similar to the procedures that are typical of scientific research [Dasgupta and David, 1994], where the production of technological knowledge appears to be governed by sets of "open knowledge institutions" (e.g., peer reviewed journals).

ECONOMIC APPROACH

Return on "Investment" for Individuals

The craving for recognition among fellow programmers has led to one of the most common misconceptions about open source projects, which is that individuals contribute to OSS projects "*only for peer-recognition, and they do not expect economic returns"* for their efforts. Some open source project participants earn direct monetary benefits in exchange for their participation. For example, a survey of 141 Linux developers found that more than 43% received some form of income, while 20% receive a salary "for their Linux programming on a regular basis" [Hertel et al., 2003]. Sometimes participants decide to contribute to an open source project in order to advertise a value-added service or a complementary, but proprietary, product that they produce (e.g., Larry Wall does Perl consulting). Finally, many individual programmers participate in open source projects with the *"expectations of future returns*," which can be classified as follows:

Help with their own projects: The reasons behind this expectation are similar to those in an old-fashioned "barn raising", where neighbors get together to build a barn in expectation that the neighbor whose barn was raised will also help them when it's their turn to build a barn [Green, 2002].

Signaling: The particular way in which open source projects are managed, and especially how contributions are attributed to individual developers, allows the best programmers to create a signal. For example, a public *changelog* file exists with the Linux project that lists programmers who have contributed to the official source and their specific inputs [Moon and Sproull, 2000; Raymond, 2001]. This is an honor and a sign of expertise among programmers [Lee et al., 2003]. By distinguishing themselves apart, individual developers can turn this signal into monetary rewards [Lerner and Tirole, 2002; Johnson, 2002; Mustonen, 2002]. However, signaling for career enhancement [Hann et al., 2004] as an incentive to participate in an open source project is not well supported by empirical evidence. For example, intrinsic motivators (e.g., creativity, skills improvement, having fun) were found to be more important than extrinsic benefits (e.g., better jobs, career advancement) for volunteer private software developers [Lakhani and Wolf, 2003; Bonaccorsi and Rossi, 2002]. Researchers also find sup*port for career* expectations and the recreational value from software development as major motivators to participate in open source projects [Hann et al., 2004].

Return on "Investment" For Organizations

Unlike individual programmers, organizations choose to participate in open source software development projects based largely on economic reasons. In many cases these organizations sell complementary applications for open source platforms and earn significant profits from these transactions [Lerner and Tirole, 2002; Lerner and Tirole, 2001]. For example, IBM promotes Linux because, among other things, it develops and sells proprietary applications for Linux [Economides and Katsamakas, 2004]. Other organizations sell accessories and complementary products such as t-shirts, books and computers with open source applications preinstalled [Varner, 1999]. Some organizations recover their cost of investment in open source initiatives by

using the project to advertise their core products and services. For example, Digital Creations use their "ZOPE" (Z Object Publisher Environment) to advertise web site creation services and Netscape used the "Mozilla" browser to advertise their web servers and web portals. Other companies give away their software for free, but sell customer support, documentation and other complimentary services [Varner, 1999]. In traditional proprietary software, a major percentage of the revenue is from software sales, and a smaller percentage is from support. An open source model dramatically shifts this to 0% sales and 100% support. With this approach, the free software is supposed to attract more users, and therefore more support revenues. Capturing even a small fraction of these users for complimentary OSS services would provide significant financial gains to organizations selling such services.

"Public Goods" Approach

Some researchers classify OSS as a public good supplied by voluntary private contributions [Bitzer and Schröder, 2002] since it is non-excludable and nearly non-rival. Furthermore, OSS provides benefit to society by increasing competition within the software industry. However, participants in open source projects apparently do not receive enough benefits *to encourage everyone* who is skilled enough to participate in the project. If this scenario is modeled as a game, then a self-interested individual would consider providing such software only if the benefits gained justified the cost of programming. Nevertheless, each developer/user is tempted to free ride and waits for others to develop the software instead. Therefore, one of the *Nash equilibria for the game is for all players to be free riders*, i.e., not participate in an open source project, with the result that no open source software is developed. However, many successful open source projects not only retain many of the initial developers but also attract new developers. Existing literature offers some explanations for this apparent contradictory behavior. Metaphorically, each programmer contributes a brick and *each* gets back a complete house in return. However, in software, unlike with physical goods, one person's gain does not come at the expense of another, because a copy does not deplete the original in any way. Sharing software is not a zero-sum game, and there are tremendous efficiencies from participating in such a cooperative endeavor [Prasad, 2001]. Furthermore, if the open source software project is modeled as a war of attrition with complete information, job signaling, repeated contribution to the public good, uncertainty in programming, and no delay, then the software will be provided swiftly, by young, low-cost individuals who gain considerably by signaling their programming skills [Bliss and Nalebuff, 1984; Hendricks et al., 1988; Bilodeau and Slivinski, 1996]. This resulting equilibrium suggests that the startup of an open source project displays bandwagon dynamics, i.e., the addition of one agent to the community of programmers immediately increases the probability of other non-members joining, sustaining the project over a long period.

V. OSS PROJECT ADOPTION AND DEPLOYMENT PHASE

Linux and other open source systems face the challenge of getting adopted in an environment dominated by proprietary standards. Recent developments in the theory of critical mass in diffusion of technologies with network externality may help to explain this phenomenon [Bonaccorsi and Rossi, 2002]. Another theory that takes a psychological/sociological perspective and may shed light on open source software diffusion is the *diffusion of innovation theory* [Rogers, 1995][1]. According to this theory, "*technological innovation is communicated through particular channels, over time, among the members of a social system*." It has been applied to information technology innovations, artifacts, techniques, and has been used as a theoretical basis for IS research projects. The application of this theory in the context of open source software is currently not clear. When applied to network technologies, a large part of the diffusion

[1] Please see Roger Clarke's excellent primer at http://www.anu.edu.au/people/Roger.Clarke/SOS/InnDiff.html

theory forecasts the emergence of a dominant standard and, as a second instance, the coexistence of different standards only if the diffusion dynamic of competing technologies begin at the same time. Open source software, however, spreads across systems even though there are pre-existing dominant standards and products (e.g., Apache vs. Microsoft IIS).

Adoption of open source applications by organizations can also be explained by existing theories on software adoption such as the Technology Acceptance Model [Davis, 1989] and its numerous extensions. Some of the factors found to influence adoption include perceived usefulness, ease of use, job relevance, output quality, results demonstrability, image enhancement, prior experience and subjective norms [Hartwick and Barki, 1994; Karahanna et al., 1999; Venkatesh and Davis, 2000; Moore and Benbasat, 1991; Adams et al., 1992]. Although most IT adoption studies have been conducted with commercial proprietary systems, their findings may also apply to open source projects since the findings do not suggest that the system's development methodology has an impact on the acceptance of the system.

Technology adoption and diffusion studies have been conducted in contexts similar to open source software, but they often lack characteristics found in the open source movement. For example, Chau and Tam [1997] studied the adoption of open-systems as an *internal* IS innovation and found only weak order effects on end-users and/or the underlying business process. The significant determinants towards open-systems adoption included: *perceived barriers, satisfaction with existing systems* and *compliance with standards* and *interoperability* [Chau and Tam, 1997]. Some researchers found that in software process innovations (such as object-orientated programming languages), there were greater propensities towards assimilation when there were more activities to spread learning-related costs, extensive knowledge related to the innovation, and a greater diversity of technical knowledge and activities [Fichman and Kemerer, 1997]. Nelson and Shaw [2003], in their study of adoption of interorganizational system (IOS) standards, found that the firm's *installed base, top management support, feasibility, direct network effects, mission* and *conduciveness towards interoperabilit*y were significant determinants of IOS standards diffusion. Collectively, these results (and others similar to them) provide a rich framework to understand OSS adoption, but we need to add factors such as the role of project moderators, the notion of community goods, peer recognition, and other social frameworks that are important for OSS project participation.

VI. COORDINATION MECHANISMS IN AN OSS PROJECT

An important area of research in the management of OSS projects is coordination. Understanding coordination in software projects, as in other complex tasks, has long been difficult. Several factors, including the scale of the project, uncertainty of outcome, and interdependence among project activities, make coordination of software projects challenging [Kraut and Streeter, 1995]. As a project size increases, keeping track of who knows what and successfully bringing that expertise to bear on a given project is difficult [Faraj and Sproull, 2000]. The organizational challenges faced by OSS projects are significant because the project must deal not only with problems faced by any software development process, but also with the complexity of coordinating efforts of a geographically distributed base of volunteers working on the software. The hierarchically organized and top-down planned structure of most proprietary software projects is replaced in OSS by a new kind of bottom-up, non-coercive and largely decentralized structure, even if shared behavioral rules are present. Distributed work slows development, impedes changes to the software, and leads to less communication between project participants [Herbsleb and Mockus, 2003]. Since most OSS communication is electronic, coordination could also suffer from the lack of spontaneous conversation and the inability to share complex and ambiguous messages [Yamauchi et al., 2000]. Despite these inherent weaknesses in OSS project management, we still see evidence of hierarchical coordination emerging without the support of property-rights based organization [Bonaccorsi and Rossi, 2002]. In the rest of this section we show how the literature on coordination and communication in OSS highlight the following approaches: (a) conventions, (b) economics, (c) the shared knowledge about the software code, and (d) network governance principles.

CONVENTION AS A BASIS FOR COORDINATION AND COMMUNICATION

Conventions play a major role in the way open source projects are coordinated, and issues such as acknowledging contributions, preventing forking, fixing software shortcomings, and adding new enhancements are managed. Contributions of individuals in an open source project can be gauged from records maintained by OSS project initiator(s), also referred to as "project moderators". The developer who contributes more, according to the records, is given more respect and weight in future decisions about the project [Cavalier, 1998]. This also minimizes the risk of "forking" [Kaisla, 2001], which happens when someone takes the open source code for a specific application and develops it to start a different "evolutionary" direction. In practice, however, such forking is rare and is always accompanied by re-labeling and public justification. The open source movement has an elaborate, largely spontaneous set of *ownership conventions,* which regulate who can modify the software, the circumstances under which it can be modified, and who has the right to redistribute the modified versions [Raymond, 2001]. Such conventions do not allow distributing changes to a project without the cooperation of the moderators and also prevent removal of a developer's name from a project history, credits or maintainer list without the developer's explicit permission.

ECONOMICS AS A BASIS FOR COORDINATION AND COMMUNICATION

According to The Open Group [2003], there is also an economic reason that prevents forking. Large hardware vendors (i.e., OEMs) make it clear that they do not want to go through the UNIX system wars again. Open source software projects heed these warnings because the adoption of an open source application depends on more software vendors producing products compatible with the OSS application. When an OSS application has multiple versions/flavors, software vendors are forced to choose the versions to support and incorporate into their products, leading to an adverse impact on the adoption of the OSS application.

SHARED KNOWLEDGE AS A BASIS FOR COORDINATION AND COMMUNICATION

The organizational theory of professions and the classic theory of clubs suggest two reasons why a community of open source participants is at least as good as a firm. First, the constituents of OSS projects have symmetry of absorptive capacity, i.e., the participants have similar technical skills and understanding of the project objectives. Second, the software itself is a capital structure embodying all knowledge and information necessary to take the project further [Garzarelli, 2002]. Because all developers have access to the relevant objective knowledge inherent in the source code, they need very little communication. Even this little communication takes place mostly when they are lobbying for their enhancements to be incorporated in the next version. The inherent knowledge in the software makes the communication process simple and relatively less costly. In addition, social contracts prevent conflicting interests from destroying the cooperative mode of interaction in OSS projects [Kaisla, 2001].

NETWORK GOVERNANCE AS A BASIS FOR COORDINATION AND COMMUNICATION

More recently network governance has been used to explain the coordination among open source contributors [Sagers, 2004]. Network governance is "characterized by informal social systems rather than bureaucratic structures and formal contractual relationships found in traditional firms" [Jones et al., 1997]. The restrictive access to the development team improves coordination within the project and safeguards exchanges among OSS project members [Sagers, 2004]. Further, collective sanctions safeguard exchanges among project members, and the importance of reputation among project members aids in managing conflicts within the project [Sagers, 2004].

VII. IMPACT OF OSS PROJECTS

Two dominant areas in OSS impact literature are the measurements of project performance and the impact of OSS on the software industry. Success or failure of proprietary software applications is generally measured by their market performance (e.g., installed base, market share). These measures, however, cannot be directly applied to OSS projects because of the difficulty in tracking and estimating the installation base or market share of OSS. Currently, there are more than 100,000 OSS projects [www.SourceForge.net] underway, and most are small projects with less than 50 developers involved in each. Although some researchers have focused on analyzing measures of success in open source software projects [Crowston et al. 2003], there is little published research that investigates the determinants of the success in an open source project. Stewart and Ammeter [2002] identify some software-specific characteristics such as development status of the application. However, they do not include critical factors such as software license information in their analysis. Consideration of such license data will likely complicate the model due to the potential for endogenous license choice in project success. This issue has been highlighted in another study that explores the determinants of open source project success as measured by project popularity [Sen, 2005]. A study by Sagers [2004] defines project success in terms of the quality of coordination among project members, assuming that better coordination would allow the project to be more successful.

Research on the impact of OSS on the software industry can benefit from two referent disciplines: industrial organization (IO) and economics of software development. The existing IO literature in information systems has largely focused on the importance of complementary components in the successful adoption of software, with emphasis on the implications of compatibility and the incentives of rival firms to make compatible components [Shy, 2001]. These studies generally conclude that compatibility is the unique equilibrium under most demand conditions. Farrell and Katz [2000] show that the integration of a monopolist into a competitive complementary market may weaken the innovation incentives of independent firms. Complementary components are typically assumed to be symmetric in systems literature. Some studies have also focused on the role played by positive network externalities in defining the industrial organization of software and/or hardware industry. For instance, early work in industrial organization studied the effects of network externalities on *de facto* standardization [Farrell and Saloner, 1985; Katz and Shapiro, 1992; 1986; 1985]. Some of these studies focused on the role of technology sponsorship in *creating a strategic advantage* when incompatible technologies compete in markets with network externalities [Katz and Shapiro, 1985]. Another key finding is that the strength of consumer preferences for software variety drives the equilibrium outcomes for hardware platforms. When consumers place a high value on software variety relative to the difference in value of different hardware technologies, the equilibrium outcome is a *de facto* standardization in the hardware market (resulting in the only surviving hardware that is supported by software providers) [Church and Gandal, 1992].

Most of these studies have been conducted in the context of commercial software applications. Mustonen [2003] conducted one of the few studies specifically addressing open source related issues. In this study, in which the participation in open source projects is endogenous, it is shown that low implementation cost of an open source application is crucial for its survival when it competes with a proprietary application. In other studies, Casadesus and Ghemawat [2003] examined the competition between Windows and Linux. Economides and Katsamakas [2004] compare software industry structures based on proprietary platforms with those based on open source platforms and identified the demand conditions that support various equilibrium outcomes. One of their results relevant to OSS research is that "*social welfare in the software industry may be higher when the platform is open source rather than proprietary and the cost of adoption of open source platform is small*." Another important result from the study is that the "*proprietary software industry is more profitable than the open source software industry, if the users have strong preference for application variety*." These results, however, do not take into account the coexistence of both the open source and proprietary platforms. An extension of this model

proposed by the same authors addresses this limitation and concludes that "*when systems based on an open source platform with an independent proprietary application compete with systems based on proprietary platforms and independent proprietary application, the open source system is typically dominant in terms of market share*." Sen [2005] proposes a more general model for competition between "free" open source software, commercial open source software, and commercial proprietary software. He identifies the conditions under which freely available open source software and/or the commercial version of the same will adversely affect the market position of proprietary software, and suggests some strategic steps that commercial software vendors can take in order to compete successfully.

VIII. CONCLUSION: FUTURE RESEARCH OPPORTUNITIES

Despite all of the active OSS related research, major gaps remain that must be filled to enhance our understanding of this new form of software. This section summarizes our recommendations in order to provide directions for future OSS related research. Figure 3 summarizes the various future research opportunities using our research classification framework.

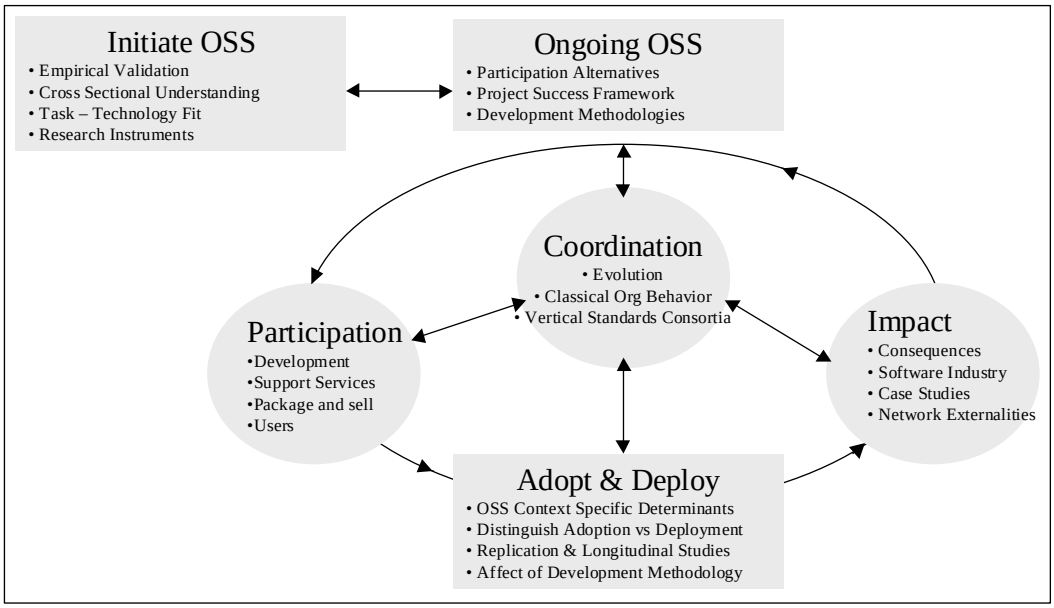

Figure 3. Future Research in Open Source Software

INITIATION PHASE

Researchers have addressed issues relating to individual and organizational motivation for initiating open source projects, mostly using qualitative case studies (e.g., studies on Linux and Apache) and/or purely analytical approaches. Empirical validation of the results in a more general setting that involves several open source projects can improve our understanding of OSS. There is also a need to develop measurement instruments for personal/psychological, technical, economic/strategic, and social/political factors that motivate individuals and organizations to initiate open source projects. We can borrow instruments from referent areas, such as Davis' 14-question instrument for perceived usefulness and ease of use [Davis, 1989] or the extensions in Venkatesh and Davis [2000]. Moore and Benbasat [1991] developed a general purpose instrument that examines eight areas in an end user's perceptions of an innovation. Modifications are still necessary, however, to make these instruments suitable in an OSS context and to validate them. Comparison of the underlying capabilities and limitations of different open source

based solutions are also important. Task-technology fit literature can provide insights into OSS adoption determinants, performance gaps, and coordination mechanisms [Goodhue and Thompson, 1995; Goodhue, 1995].

ONGOING PROJECTS

The ongoing OSS projects phase literature is dominated by qualitative and analytical approaches. There is a need for empirical studies and instruments to measure the motivational factors and to understand the relationships among these factors. Empirical investigations into the success and failure of OSS projects can also contribute in a great way. Presently, we have anecdotal evidence of success or failure for a few large open source projects. We lack a general framework that can be applied to the performance of open source projects of any size. Another rich area concerns the impact of OSS on the management of *in-house* systems development efforts and off-the-shelf applications.

ADOPTION / DEPLOYMENT

Existing technology diffusion literature provides an excellent starting point for investigating factors that lead to the adoption and deployment of open source software. Currently, there is a need for introducing and testing OSS specific contextual factors such as the role of project moderators, the notion of community goods, peer recognition, and other social framework issues. There is also a need for replication and longitudinal studies that validate the application of diffusion theories. Existing technology diffusion theories rarely examine the role of development methodology on diffusion. It would be interesting to investigate if the inclusion of the development methodology as a factor sheds different light on the adoption or deployment decisions. Researchers should also clearly distinguish between studies on *intentions to adopt, adoption,* and *deployment* of OSS.

COORDINATION

A significant opportunity exists for studying the evolution of coordination mechanisms in open source projects and the role played by open public networks (like the Internet) and online project coordination tools (e.g., Sourceforge.net). There is also a need to investigate the extent to which open source coordination mechanisms can be explained using classical theories from organizational structure, and whether the two different forms of organizations can learn about coordination and control from each other. With the emergence of organizations such as RosettaNet and the Open Geospatial Consortium, there may be a third form that provides a middle ground between the two extremes. These non-profit and predominantly *vertically oriented standards development consortia* (VSC) pool development resources among user groups and offer more formal coordination mechanisms and networks. Participation in these VSC is voluntary, decision making is consensus driven, and most are platform independent, vendor neutral and non-profit organizations [Nelson and Shaw, 2003]. Vertical standards consortia structures may provide a near-perfect coordination model for future OSS projects by offering flexibility, independence, speed and a semi-formal coordination mechanism.

OPEN SOURCE SOFTWARE IMPACT AND CONSEQUENCES

A rich source of research opportunities in OSS relates to the impact and consequences of innovation diffusion [Rogers, 1995]. The consequences can be measured in all OSS phases including initiation (product pricing, substitute products), ongoing projects (development effort, through-put), adoption (expectations, costs) and deployment (productivity, network externalities). Another impact area for future research is the examination of the impact of open source applications on the traditional commercial software industry. This could be addressed analytically using theories from industrial organization, or empirically by collecting industry level data on the performance of commercial software producers before and after the emergence of competition from open source applications. Firm level case studies could also be used to understand the strategies employed by commercial software manufacturers to combat the growing OSS threat.

RESEARCH METHODOLOGY

Existing OSS research has been dominated by software-specific case studies, such as on Linux and Apache. Results obtained from these studies are difficult to generalize because these projects could be outliers. Most open source projects have a relatively small number of participants, focus on specific functionality, and have few organizational participants. Similarly, software industrial studies are dominated by mathematical modeling, which tends to simplify the economic interactions between participating players (open source developers, commercial software producers, open source support service provides, open source sponsors and others). There is a need for more empirical research methods, both cross-sectional and longitudinal, in this area. One reason for the slow pace of empirical research is due to the large, informally structured, and geographically dispersed nature of the OSS movement.. It is difficult to collect relevant cross-sectional and/or panel data. SourceForge.net, a website that maintains information on thousands of open source projects, is a possible source for such information. Other data sources for particular open source projects could include email archives of source code change history and problem reports which are used to quantify aspects of developer participation, core team size, code ownership, productivity, defect density, and problem resolution intervals. Concurrent Version Control Archive (CVS) maintained by various open source projects can also provide a wealth of information on the projects.

REFERENCES

"Governments like open-source software, but Microsoft does not," *The Economist*, September 11, 2003.

"Open Standards- Open Source: The business, legal, and technical challenges ahead," *Proceedings The Open* Group, June 24-25, 2003, Minneapolis, MN, USA, pp. 18.

Adams, D.A., R.R. Nelson, and P.A. Todd (1992) "Perceived Usefulness, Ease of Use, and Usage of Information Technology: A Replication," *MIS Quarterly,* (16)2, pp. 227-247.

Allen, R.C. (1983) "Collective Invention," *Journal of Economic Behavior and Organization*, (4), pp. 1-24.

Benson, P. et al. (1980) "Interpersonal correlates of nonspontaneous helping behavior," *Journal of Social Psychology*, (110)1, pp. 87-95.

Bilodeau, M. and A. Slivinski (1996) "Toilet cleaning and department chairing: volunteering a public service," *Journal of Public Economics*, (59), pp. 299-308.

Bitzer, J. and P.J.H. Schröder (2002) "Bug fixing and Code writing: The private provision of open source," *Discussion Paper, German Institute for Economic Research,* Berlin.

Blau, P. (1964) *Exchange and Power in Social Life*, New York, NY: Wiley.

Bliss, C. and B. Nalebuff (1984) "Dragon Slaying and ballroom dancing: the private supply of public good," *Journal of Public Economics*, (25), pp. 1-12.

Bonaccorsi, A., and C. Rossi (2002) "Why open source can succeed?" *LEM Working Paper*, July.

Casadesus, R.M. and P. Ghemawat (2003) "Dynamic mixed duopoly: a model motivated by Linux vs. Windows," *Harvard Business School Working Paper*.

Cavalier, F.J. III. (1998) "Some implications of Bazaar size," *Mib Software Working Paper*.

Chau, P. and K. Tam (1997) “Factors Affecting Adoption of Open Systems: An Exploratory Study,” *MIS Quarterly*, (21)1, pp. 1-24.

Church, J. and N. Gandal (1992) “Network effects, software provision, and standardization,” *Journal of Industrial Economics*, (40)1, pp. 85-103.

Clary, E. G. and L. Orenstein (1991) “The amount and effectiveness of help: the relationship of motives and abilities to helping behavior,” *Personality and Social Psychology Bulletin*, (17), pp. 58-64.

Clary, E. G. et al. (1998) “Understanding and assessing the motivations of volunteers: A functional approach,” *Journal of Personality and Social Psychology*, (74)6, pp. 1516-1530.

Crowston, K., Annabi, H., and Howison, J. (2003) “Defining Open Source Software Project Success,” *Proceedings of the 24th International Conference on Information Systems,* Dec 12-15 2004, Seattle, WA.

Dasgupta, P. and P.A. David (1994) “Toward a new economics of science,” *Research Policy*, (23), pp. 487-521.

Davenport, T.H., and M.C. Beers (1995) “Managing information about processes,” *Journal of Management Information Systems*, (12)1, pp. 57-80.

Davis, F.D. (1989) “Perceived Usefulness, Perceived Ease of Use and User Acceptance of Information Technology,” *MIS Quarterly,* (13)3, pp. 319-340.

Economides, N. and E. Katsamakas (2004) “Two-sided competition of proprietary vs. open source technology platforms for the software industry,” *NET Institute Working Paper #04-22*, August.

Faraj, S. and Sproull, L. “Coordinating expertise in software development teams,” *Management Science*, (46)12, 2000, pp. 1554-1568.

Farrell, J. and Katz, M.L. “Innovation, rent extraction, and integration in systems markets,” *Journal of Industrial Economics*, 48(4), 2000, pp. 413-432.

Farrell, J. and G. Saloner (1985) “Standardization, compatibility, and innovation,” *Rand Journal of Economics*, (16), pp.70-83.

Fichman, R. and C. Kemerer (1997) "The Assimilation of Software Process Innovations: An Organizational Learning Perspective," *Management Science*, (43)10, pp. 1345-1363.

Garzarelli, G. (2003) “Open source software and the economics of organization,” in J. Birner and P. Garouste (eds.) *Markets, Information and Communication: Austrian Perspectives on the Internet Economy*, New York, NY: Routledge, pp. 47-62.

Goodhue, D.L. and R.L. Thompson (1995) "Task-Technology Fit and Individual Performance," *MIS Quarterly,* (19)2, pp. 213-237.

Goodhue, D.L. (1995) "Understanding user evaluations of information systems," *Management Science,* (41)12, pp. 1827-1845.

Green, E. L. (2002) “Economics of Open Source,” http://badtux.org/home/eric/editorial/economics.php (December 2002)

Hackman, J.R. and G. Oldham (1980) *Work ReDesign,* Reading, MA: Addison-Wesley.

Hann, Il-Horn, J. Robert, and S.A. Slaughter (2004) "Why developers participate in open source software projects: An empirical investigation," *Proceedings of the 25th International Conference on Information Systems*, 13-15th Dec 2004, pp. 821-830.

Hartwick, J. and H. Barki (1994) "Explaining the Role of User Participation in Information System Use," *Management Science,* (40)4, pp. 440-465.

Hendricks, K., A. Weis, and C. Wilson (1988) "The war of attrition in continuous time with complete information," *International Economic Review*, (29), pp. 663-680.

Herbsleb, J. and A. Mockus (2003) "An empirical study of speed and communication in globally-distributed software development," *IEEE Transactions on Software Engineering*, 29(6), pp. 481-494.

Hertel, G., S. Niedner, and S. Herrmann (2003) "Motivation of software developers in open source projects: An Internet-based survey of contributors to the Linux kernel," *Research Policy*, (32), pp. 1159-1177.

Johnson, J.P. (2002) "Open source software: Private provision of a public good," *Journal of Economics and Management Strategy*, 11(4), pp. 637-662.

Jones, C., W.S. Hesterly, and S.P. Borgatti (1997) "A general theory of network governance: exchange conditions and social mechanisms," *Academy of Management Review*, (22)4, pp. 911-945.

Kaisla, J. (2001) "Constitutional Dynamics of the open source software development," Working Paper, Department of Industrial Economics and Strategy, Copenhage Business School, May 2001.

Karahanna, E., D.W. Straub, and N.L. Chervany (1999) "Information Technology Adoption Across Time: A Cross-Sectional Comparison of Pre-adoption and Post-adoption Beliefs," *MIS Quarterly,* (23)2, pp. 183-213.

Katz, M.L. and C. Shapiro (1992) "Product introduction with network externalities," *Journal of Industrial Economics*, (75), pp.424-440.

Katz, M.L. and C. Shapiro (1986) "Technology adoption in the presence of network externalities," *Journal of Political Economy*, (75), pp.424-440.

Katz, M.L. and C. Shapiro (1985) "Network Externalities, competition, and compatibility," *American Economic Review*, (75), pp.424-440.

Koch, C. (2003) "Your Open Source Plan," *CIO*, March 15.

Kraut, R.E. and L.A. Streeter (1995) "Coordination in software development," *Communications of the ACM*, (38)5, pp. 69-81.

Lakhami, K. and E. von Hippel (2003) "How open source software works: 'free' user-to-user assistance," *Research Policy*, (32)6, pp. 923.

Lakhani, K and R.G. Wolf (2003) "Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects," MIT Sloan Working Paper No. 4425-03, September 2003, http://ssrn.com/abstract=443040.

Lee, S., N. Moisa, and M. Weiss (2003) "Open source as a signaling device - An economic analysis," *WP Series Finance and Accounting,* No. 102, March 2003.

Lerner, J. and J. Tirole (2001) "The open source movement: key research questions," *European Economic Review*, (45), pp. 819-826.

Lerner, J. and J. Tirole (2002) "Some Simple Economics of Open Source," *Journal of Industrial Economics*, (50), pp. 197-234.

Mauss, M. (1967) *The Gift: Forms and functions of exchange in archaic societies*, New York: Norton

Moon, J.Y. and L. Sproull (2000) "Essence of Distributed Work: The Case of the Linux Kernel" *Firstmonday*, (5)11, November 2000. http://www.firstmonday.org/issues/issue5_11/moon/index.html.

Moore, G.C. and I. Benbasat (1991) "Development of an Instrument to Measure The Perceptions of Adopting an Information Technology Innovation," *Information System Research,* (2)3, pp. 192-222.

Mustonen, M. (2002) "Why do firms support the development of substitute copyleft programs?" FPPE, University of Helsinki, mimeo.

Mustonen, M. (2003) "Copyleft - The Economics of Linux and Other Open Source Software," *Information Economics and Policy*, 15(1), pp. 99-121.

Nelson, M. and M. Shaw (2003) "The Adoption and Diffusion of Interorganizational System Standards and Process Innovations," *MISQ Special Issue Workshop, Standard Making: A Critical Research Frontier for Information Systems,* Seattle, WA., December 2003.

Nuvolari, A. (2003) "Open source software development: some historical perspectives," *Eindhoven Centre for Innovation Studies, The Netherlands, Working Paper 03.01*, January 2003.

Prasad, G. (2001) "Open Source Economics: Examining some pseudo-economic arguments about open source," *http://linuxtoday.com/infrastructure/2001041200620OPBZCY* (April 12, 2001).

Raymond, E.S. (2001) *The Cathedral and the Bazaar*, Sebastopol, CA: O'Reilly.

Robles, G. (2004) "A Software Engineering Approach to Libre Software," http://www.opensourcejahrbuch.de/2004/pdfs/III-3-Robles.pdf.

Rogers E.M. (1995) *Diffusion of Innovations*, 4th Edition (originally published in 1962), New York, NY: The Free Press.

Rousseau, D. (1995) *Psychological Contracts in Organizations: understanding written and unwritten agreements*, Thousand Oaks, CA: Sage Publications.

Sagers, G.W. (2004) "The influence of Network governance factors on success in open source software development projects," *Proceedings of the 25th International Conference on Information Systems*, 13-15th Dec 2004, pp. 427-438.

Sen. R. (2005) "Open Source Software Development Projects: Determinants of Project Popularity," *http://econwpa.wustl.edu:8089/eps/em/papers/0510/0510003.pdf.*

Sen, R. (2005) "A Strategic Analysis of Competition Between Open Source and Proprietary Software," *Industrial Organization 0510004, Economics Working Paper Archive at WUSTL*.

Shy, O. (2001) *The Economics of Network Industries*, Cambridge, UK: Cambridge University Press, pp 13-42.

Stewart, K. J., and T. Ammeter (2002) "An Exploratory Study of Factors Influencing the Level of Vitality and Popularity of Open Source Projects," *Proceedings 23rd International Conference on Information Systems*, pp. 853-857.

Varner, P.E. (1999) "The economics of open source," *http://www.cs.virginia.edu/~pev5b/writing/econ_oss/*.

Venkatesh, V. and F.D. Davis (2000) "Theoretical Extension of The Technology Acceptance Model: Four Longitudinal Field Studies," *Management Science,* (46)2, pp. 186-204.

Yamauchi, Y., et al. (2000) "Collaboration with lean media: how open source software succeeds," *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work, pp. 329-338*.

Appendix A

Strong Copyleft Licenses provide that once a program is licensed by a developer, the subsequent programs based on the original must also be licensed similarly [e.g. GPL]. Thus, for software with a Strong Copyleft license, the freedom to re-distribute the software is low.

Weak Copyleft Licenses provide that once a program is licensed by a developer, the subsequent programs based on the original must also be licensed similarly, however they can be released under a different license under certain conditions [e.g. LGPL]. Thus, under Weak Copyleft license, the end-user has moderate freedom to re-distribute the software.

Non-copyleft licenses: Developers are not obligated to inherit the original license [e.g. BSD]. This type of license provides the maximum freedom to the users to redistribute the open source software.

License Types of Select Open source Software

License	**Strong Copyleft**	**Weak Copyleft**	**Non-Copyleft**
GNU General Public License	X		
GNU Lesser General Public License		X	
License of Guile		X	
License of the run-time units of the GNU Ada compiler		X	
X11 License (MIT License)			X
Expat License			X

License	Strong Copyleft	Weak Copyleft	Non-Copyleft
(MIT License)			
Standard ML of New Jersey Copyright License			X
Public Domain			X
Cryptix General License			X
Modified BSD license			X
University of Illinois/NCSA Open Source License			X
License of ZLib			X
License of the iMatix Standard Function Library			X
W3C Software Notice and License			X
Berkeley Database License (aka the Sleepycat Software Product License)			X
OpenLDAP License, Version 2.7			X
License of Python 1.6a2 and earlier versions			X
License of Python 2.0.1, 2.1.1, and newer versions			X
License of Perl			X
Clarified Artistic License			X
Zope Public License version 2.0			X
Intel Open Source License (as published by OSI)			X
License of Netscape Javascript		X	
eCos license version 2.0		X	
Eiffel Forum License, version 2			X
License of Vim, Version 6.1 or later		X	
Boost Software License			X
EU DataGrid Software License			X
The license of Ruby			X
XFree86 1.1 License			X
Affero General Public License	X		
The Condor Public License			X
Original BSD license			X

License	Strong Copyleft	Weak Copyleft	Non-Copyleft
OpenSSL license	X		
Academic Free License, version 1.1.			X
Open Software License, version 1.0		X	
Apache License, Version 1.0			X
Apache License, Version 1.1			X
Apache Software License, version 2.0			X
Zope Public License version 1			X
License of xinetd	X		
License of Python 1.6b1 and later versions, through 2.0 and 2.1			X
Old OpenLDAP License, Version 2.3			X
IBM Public License, Version 1.0			X
Common Public License Version 1.0			X
Eclipse Public License Version 1.0			X
Phorum License, Version 2.0			X
LaTeX Project Public License			X
Mozilla Public License (MPL)		X	X
Common Development and Distribution License (CDDL)		X	
Netizen Open Source License (NOSL), Version 1.0		X	
Interbase Public License, Version 1.0		X	
Sun Public License		X	
Nokia Open Source License		X	
Netscape Public License (NPL)		X	
Jabber Open Source License, Version 1.0			X
Sun Industry Standards Source License 1.0		X	
Q Public License (QPL), Version 1.0			X
PHP License, Version 3.0			X
Zend License, Version 2.0			X

License	Strong Copyleft	Weak Copyleft	Non-Copyleft
Vita Nuova Liberal Source License	X		
Lucent Public License Version 1.02 (Plan 9 license)			X
Apple Public Source License (APSL), version 2			X
GNU Free Documentation License	X		
FreeBSD Documentation License			X
Apple's Common Documentation License, Version 1.0			X
Open Publication License, Version 1.0		X	
Public Domain		X	
Artistic License 2.0			
BSD License			X
GNAT Modified GPL (GMGPL)		X	
MIT/X Consortium License			X
MITRE Collaborative Virtual Workspace License (CVW)	X		
Open Software License	X		
Ricoh Source Code Public Licence	X		
The CeCILL License			X
The Clarified Artistic License			X
The Latex Project Public License (LPPL)			X
The Open Content License	X		
W3C License			X
zlib/libpng License			X
Zope Public License (ZPL)			X

ABOUT THE AUTHORS

Matthew Nelson is an Assistant Professor in Information Systems. He obtained his Ph.D. in Business Administration from the University of Illinois at Urbana-Champaign in 2003. His research interests include technology diffusion, standards development organizations, and the value of information technology.

Ravi Sen is an Assistant Professor at Mays Business School, Texas A&M University. He obtained his PhD in Business Administration from University of Illinois at Urbana-Champaign in 2003. His research interests include online search behavior, economics of information search, open source software development and e-security.

Chandrasekar Subramaniam is an Assistant Professor in Information Systems at the University of North Carolina at Charlotte. He obtained his Ph.D. in Business Administration from the University of Illinois at Urbana-Champaign in 2003. His research interests include impact of electronic commerce, value of information technology, and inter-organizational information systems.

EDITOR'S NOTE: The following reference list contains the address of World Wide Web pages. Readers, who have the ability to access the Web directly from their computer or are reading the paper on the Web, can gain direct access to these references. Readers are warned, however, that

1. these links existed as of the date of publication but are not guaranteed to be working thereafter.

2. the contents of Web pages may change over time. Where version information is provided in the References, different versions may not contain the information or the conclusions referenced.

3. the authors of the Web pages, not CAIS, are responsible for the accuracy of their content.

4. the author of this article, not CAIS, is responsible for the accuracy of the URL and version information.

.

Communications of the Association for Information Systems

ISSN: 1529-3181