A framework for creating hybrid-open source software communities

Srinarayan Sharma, Vijayan Sugumaran & Balaji Rajagopalan

Department of Decision and Information Sciences, School of Business Administration, Oakland University, Rochester, MI 48309, USA, email: srisharm@oakland.edu, sugumara@oakland.edu, rajagopa@oakland.edu

Abstract. *The open source software (OSS) model is a fundamentally new and revolutionary way to develop software. The success of the OSS model is also setting the stage for a structural change in the software industry; it is beginning to transform software industry from manufacturing to a service industry. Despite the success of the OSS model, for-profit organizations are having difficulty building a business model around the open source paradigm. Whereas there are some isolated empirical studies, little rigorous research has been done on how traditional organizations can implement and benefit from OSS practices. This research explores how organizations can foster an environment similar to OSS to manage their software development efforts to reap its numerous advantages. Drawing on organizational theory, we develop a framework that guides the creation and management of a hybrid-OSS community within an organization. We discuss the implications of this framework and suggest areas for future research.*

Keywords: open source software, software development, hybrid-OSS community, OSS framework, features of OSS

INTRODUCTION

The open source software (OSS) model is a fundamentally new and revolutionary way to develop software (DiBona *et al*., 1999; Moody, 2001; Raymond, 1999). In the OSS approach, source code of the product is made freely available to anyone to view, modify and distribute under open source definition compliant licence, as articulated under the open source initiative (http://www.opensource.org). In the software industry, which is struggling to find ways of developing quality software products, the OSS development approach has helped produce reliable, high quality software quickly and inexpensively; (Harvard Business Review, 2000; Mockus *et al*., 2000; The Economist, 2001a) by addressing many aspects of the 'software crisis' (Feller & Fitzgerald, 2000). Besides, it offers the potential for a more flexible technology, quicker inno-

vation, and lower cost (Plotkin, 1998; Deckmyn, 2000; Portelli, 2000; Borrell, 2001). OSS has, arguably, helped companies achieve greater penetration of market and offered opportunity to establish an industry standard and, thus, increased competitive advantage over its competitors (Plotkin, 1998; Portelli, 2000; Borrell, 2001). It has also helped build developer loyalty as developers feel empowered and a sense of ownership of the product (Portelli, 2000). Some traditional closed-source vendors are developing strategies to reverse the tide of open source software movement (*The Economist*, 2001b; Hilson, 2001; Weiss, 2001). However, the future is more likely to be reflective of companies using the open source model and taking advantage of the unique opportunities such as the broad developer base and much needed user input that the OSS model provides (Plotkin, 1998; Connolly, 2001; Sullivan, 2001; Yager, 2001). With little or no marketing, open source software is finding its way into the information technology (IT) shops of a variety of companies and has been able to gain dominant market shares in several categories for many classes of business applications (Borrell, 2001; Sullivan, 2001).

Despite the success of the OSS model, for-profit organizations are having difficulty building a business model around the open source paradigm (Portelli, 2000). Companies like Collab.Net are springing up to help software companies alleviate some of the problems associated with use of the open source model by bringing sponsors and developers together. They help established companies like Sun Microsystems or Hewlett-Packard (HP) launch an open source product and support open source collaborative development (Borrell, 2001). Technology powerhouses such as HP, Intel, IBM, NEC, etc. are helping create an Open Source Development laboratory to promote open source software collaboration and growth (Weiss, 2000; 2001). Based on some of the high-profile success stories, OSS proponents argue that quality software can be produced in a relatively short period of time, with very little cost, by some of the best programmers in the profession. However, a counter argument can also be made that not all OSS initiatives have been successful, for example, SourceXchange and Eazel (Feller and Fitzgerald, 2002). Nevertheless, empirical evidence is beginning to emerge that establishes the viability and effectiveness of the OSS development paradigm (Plotkin, 1998; The Economist, 2001a). Whereas there are some isolated empirical studies (Mockus *et al*., 2000), little rigorous research has been done on how traditional organizations can implement and benefit from OSS practices (Feller & Fitzgerald, 2000; Mockus *et al*., 2000). This research explores how organizations can foster an environment, similar to OSS, to manage their software development efforts to reap their numerous advantages. We argue that such an attempt aimed at imbibing specific features of the open source model in a traditional organization will result in the creation of a hybrid-OSS environment that is better prepared to meet its changing needs. To this end, we propose a framework that guides the creation of hybrid-OSS communities in traditional organizations.

The remainder of this paper is organized as follows. In the next section, we analyse OSS organizations to comprehend the process of creating and sustaining OSS communities. Based on our understanding of OSS communities, in *Framework for creating hybrid-OSS communities*, we provide a framework for organizations to foster a hybrid-OSS environment to manage their software development efforts within existing constraints. This is followed by a discussion

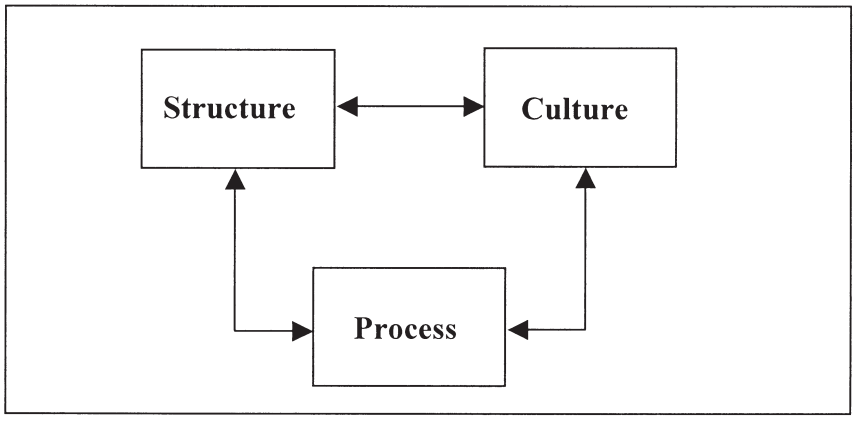

Figure 1. Framework for analysing organizations.

of the implications of this framework in *Discussion*. Finally, we present concluding remarks and directions for future research in *Conclusions*.

THE OSS MODEL: STRUCTURE, PROCESSES AND CULTURE

To examine the OSS environment, we use an established theoretical framework based on dimensions of structure, process and culture in the organizational theory literature (Figure 1) (Galbraith, 1973; Miles and Snow, 1978; Robey, 1991). This framework emphasizes the need to pay attention to all three dimensions individually and their interactions with one another to design a sound organization.

Basic principles of the OSS model are articulated in numerous works by its proponents (see Raymond, 1999; 2001; *The Apache Software Foundation*, 2001a; 2001b; Cook, 2001; *Linux Documentation Project*, 2001; Masum, 2001) and detailed case studies of many open source software development projects (see Aoki *et al*., 2001; Mocus *et al*., 2000; Scacchi, 2001). In this section, we examine open source communities along the aforementioned dimensions of structure, processes and culture, to understand how they function. We draw on established literature in organizational theory and organizational behaviour to understand different aspects of OSS communities.

OSS organizational structure

First, we examine the structure of OSS communities along the dimensions of division of labour, co-ordination mechanisms, distribution of decision-making authority and organizational boundary. These dimensions have been widely used to analyse traditional organizations (March and Simon, 1958; Mintzberg, 1971; Nohria, 1995).

Division of labour

OSS communities are fundamentally different from traditional organizations. They consist of a large number of volunteer developers who make contributions either individually or part of a temporary team. Unlike traditional organizations, projects in these communities are not dictated by any formal plan, schedule or list of deliverables (Mockus *et al*., 2000; Raymond, 2001; Schmidt and Porter, 2001). Work is not assigned to developers; instead, they choose what to work on.

Co-ordination mechanisms

Developers in OSS communities are geographically distributed and rarely meet face-to-face. Also, they cannot devote large blocks of time to the project in a consistent manner. These conditions require them to use co-ordination mechanisms that emphasize decentralized workspaces and asynchronous communication (Asundi, 2001; Fielding, 1999; Mockus *et al*., 2000). Teams are formed to address specific problems and are disbanded when the problem is solved. Individuals on these teams work under peer supervision. Usually a core group (in the case of Apache, the Apache Group, for example) provides broad oversight of and strategic direction for these teams.

Distribution of decision-making

OSS communities have established processes for decision-making. Decision rights are primarily vested in individuals and most decisions are reached by consensus (Fielding, 1999; Markus *et al*., 2000; Mockus *et al*., 2000). With the use of E-mail, chat rooms and other information technologies that support asynchronous communication, a consensus is reached. Usually, anyone on the mailing list can vote but only votes cast by the members of the core group are considered binding.

Organizational boundaries

Unlike traditional organizations, OSS communities do not have well defined boundaries. Membership in the community is fluid; current members can leave the community and new members can also join at any time. Relationships with other organizations are formed and discarded as needed. By remaining open to new contributors, the group has an unlimited supply of innovative ideas (Fielding, 1999; Raymond, 2001).

Informal structure

In the OSS communities, there is no formal organizational structure. Any semblance of structure keeps changing with time and needs. The changes are dependent on actions taken by

volunteer developers. Most of the tasks get done informally, although under the overall direction of the core group.

Political structure

Given a lack of formal organizational structure, and its fluidity, political coalitions are built and abolished along issues of concern to the community.

Legitimate basis of authority

The OSS community is based on meritocracy (Fielding, 1999; Masum, 2001; Raymond, 2001; Schmidt and Porter, 2001). Reputation is established through quality contributions on a consistent basis that can lead to recognition and leadership roles, and is the only basis of authority in the community. Table 1 summarizes structural characteristics of open source communities. Having discussed the structural aspects of OSS communities, we describe OSS processes next.

OSS processes

The major processes within OSS communities can be classified into governance and software development (Markus *et al*., 2000; Mockus *et al*., 2000; Cook, 2001; Raymond, 2001), which are described below.

Governance process

A salient feature of OSS communities is their self-governance (Markus *et al*., 2000; Cook, 2001; Raymond, 2001). In general, the initial software developer maintains a lead role, however, formal authority is vested in a team. Projects are partitioned by lead architects or designers into manageable units/modules and handled by individuals or teams. Co-ordination of teams is the responsibility of lead architects. Further decomposition of modules may occur and module leaders may solicit inputs from members, but have the final say in cases of disputes. OSS communities typically have a central person or a group that is responsible for 'official' releases and distribution.

OSS communities appear to be using the following four different governance mechanisms: (a) membership management; (b) rules and institutions; (c) monitoring and sanctions; and (d) reputation (Markus *et al*., 2000; Cook, 2001). We briefly describe each of these mechanisms below:

Membership management: membership is usually open to anyone who is willing to participate (Open Source Foundation, 2001). A process of vetting and quality control is used for appointment to a responsible position (Markus *et al*., 2000). Community members are allowed to work on a project for a certain duration, during which their quality is assessed.

Table 1. A comparison of organizational structures between OSS communities and traditional organizations (adapted from Nohria, 1995)

	Traditional organizational forms				
	Functional	Divisional	Matrix	OSS community	Transition steps to hybrid-OSS
Division of labour	By inputs	By outputs	By inputs and outputs	By choice and knowledge	Voluntary, skill, competence, and knowledge-based task assignment
Co-ordination mechanisms	Hierarchical supervision, plans, and procedures	Divisional general manager and corporate staff	Dual reporting relationships	Membership management, rules and institutions, monitoring and sanctions and reputation	Peer-based monitoring and reputation-based membership management and rules and institutions
Decision rights	Highly centralized	Separation of strategy and execution	Shared	Highly democratic and decentralized	Consensus-based, decentralized decision-making.
Boundaries	Core/periphery	Internal/external markets	Multiple interfaces	Porous and changing	Permeable, governed by organizational rules and institutions
Importance of informal structure	Low	Modest	Considerable	High	Informal networks
Politics	Inter-functional	Corporate-division and inter-divisional	Along matrix dimensions	Shifting coalitions	Issues of concern to organizational members
Basis of authority	Positional and functional expertise	General management responsibility and resources	Negotiating skills and resources	Reputation	Reputation, based over a sufficiently long period

Formal membership is conferred by a consensus vote of the core group (Mockus *et al*., 2000).

Rules and institutions: OSS communities create and abide by a set of rules and norms. These rules are modified as the project matures over time to meet its unique requirements. Whereas general membership is open to the public, new members to the core group are added only when a frequent contributor is nominated by one member and unanimously approved by the voting members (Fielding, 1999).

OSS communities have voting systems that require only a subset of the group to be involved in any decision. This system allows OSS developers, all of whom have full-time jobs, to participate in the project. Each decision made requires a minimum number of votes, which also enforces a high degree of peer review. However, during periods of rapid and focused development, voting may become a barrier and a source of friction among developers.

Monitoring and sanctions: whereas membership in open source projects is open to any willing contributor, OSS communities manage memberships in conjunction with rules and institutions, and monitoring and sanctions. They have established means of observing behaviour and ensuring compliance. Members of the community are sanctioned if they misbehave or disrupt the progress of the project. There is social pressure against anyone who does not comply with the norms of the community. Sanctions are in the form of flaming, spamming and shunning (Markus *et al*., 2000; Raymond, 2001). Often, sanctioned members change their behaviour or leave the community. Even leaders are subject to such sanctions.

Reputation: building and maintaining reputation is one of the prime motivators for the OSS developers (Markus *et al*., 2000; Masum, 2001; Raymond, 2001). While motivation to build reputation brings in new members, a desire to maintain reputation motivates them to complete tasks on time, make quality contributions and keep projects on track.

Development process

Typically in an OSS project, developers iterate through a common series of actions while working on the software source. The development process involves the following activities (O'Reilly, 1999; Mockus *et al*., 2000; Scacchi, 2001; Schmidt and Porter, 2001):

Problem discovery: problems are reported and discussed using the following means: (a) developer E-mail list; (b) problem reporting system; and (c) USENET newsgroup. Problems on the mailing list get the highest priority and the attention of all the active developers. An agenda file with a list of high priority problems, open issues and release plans is stored in each product's repository to keep track of project status.

Finding volunteers: once the problem is discovered, volunteers are found to work on the problem. Volunteers prefer to work on problems that are related to the areas they are familiar with and have been working on. New developers work in areas in which former developers are no longer interested, or in the development of new architectures and features.

Solution identification: after having found volunteers to work on a problem, the next step is to identify the solution. Usually, many alternative solutions are available. Developers choose

solutions for their generality and portability. The chosen alternative is posted to the developer mailing list for feedback before it is implemented.

Code development and testing: once the solution has been identified, code is developed. The developer makes changes to a local copy of the source code, and tests the changes in his or her own environment.

Code change review: the tested solution is posted to the developer mailing list for review. Individual developers on the list further test this solution. If they find any problems with the solution, they suggest improvements to the originator. After a careful review, the originator makes changes to the code and again tests the solution and posts the improved solution on to the list. The process is repeated until it is approved.

Code commit and documentation: once the tested solution is approved by the list, it can be committed to the source by any of the developers, although it is preferred that the originator of the change performs the commit. Each commit results in a summary of changes being automatically posted to the Concurrent Version Control System (CVS) mailing list. All the members of the core group review the changes to ensure that changes are appropriate. Changes are also reviewed by developers outside the core group.

Release management: a core group member volunteers to serve as the release manager as the project nears a product release. The release manager identifies outstanding problems and their solutions and makes suggested changes. The role of release manager is rotated among the members of the core group.

Table 2 summarizes the characteristics of the OSS processes. Having discussed the processes of OSS communities, we describe its culture in the following section.

OSS culture

We used Schein's framework to understand the culture of OSS communities (Schein, 1984; 1996). According to this framework, the culture of organizations can be understood by examining their artifacts, values and core assumptions. Artifacts include the physical characteristics such as dress/attire and décor; mission statements, memos and slogans; and implicit communicators such as rites and rituals (Howard, 1998). Espoused values represent the conscious strategies and goals. They also represent organizational standards or criteria adopted for selection among decision-making alternatives. The central values in organizations are those that deal with transactions or events and the rules governing them. Finally, the heart of the culture is mirrored by the underlying core assumptions such as trust and loyalty, probably the most difficult to discern. Now we delineate the culture of OSS communities by examining their artifacts, values and core assumptions.

Artifacts

In the OSS model, the artifacts include electronic communication and a multicultural community. Electronic communication establishes everyone at the same level and allows people from different geographic regions and cultures participate in the open source project. The classic

Table 2. A comparison of processes between OSS communities and traditional organizations

	Traditional organization	OSS community	Transition steps to hybrid-OSS
Governance	Enforce governance	Self-Governance	Empowerment
Membership management	Management enforced Static/solid No such a thing	Community-based Fluid, but stable Professional identity	Select qualified people, assign tasks
Rules and institutions	Management makes and changes the rules	Community members make and change the rules	Facilitate creation of rules and norms, provide autonomy
Monitoring and sanctions	Monitoring of performance and behaviour kept confidential by management	Monitoring of performance and behaviour visible to everyone	Support open, peer monitoring
	Financial penalty, demotion, and layoff enacted by management	Sanctions by flaming, spamming, and shunning; expulsion Building and maintaining reputation a prime motivator	Provide authority to enforce sanctions and rewards
Reputation	No emphasis on building reputation	Loss of reputation a motivating factor	Recognize quality of work and promote reputation
Development	Survey Study Definition Configuration Procurement Design Construction Delivery	Problem discovery Finding volunteers for tasks Solution identification Code development and testing Code change review Code commit and documentation Release management	Idea generation; opportunity identification Assign tasks to people with appropriate background Move away from current development methodology Support independent code development and testing Provide repository management and project co-ordination infrastructure Allocate resources for product release and support

example is the Apache project in which the core developers are located in the USA, Britain, Canada, Italy and Germany (Fielding, 1999).

Values

OSS community members value altruism, reciprocity and gift giving, reputation and ideology highly (Perkins, 1999; Markus *et al*., 2000; Raymond, 2001). Although they are motivated by the personal benefit of using an improved software product, financial reward does not seem to be that important. They value fairness, transparency and consensus in decision-making. As a consequence, much of the OSS work is co-ordinated in the open and visible environment of the Internet, by which one's performance can be monitored by other members of the society. There is no individual ownership of products, rather, recognition of expertise is important. They believe in shared risks, shared rewards and shared ownership; (Yamauchi *et al*., 2000; Raymond, 2001).

Gaining or enhancing reputation through participation in open source projects can lead to tangible rewards, such as employment opportunities or access to venture capital to start a new company (Lerner & Tirole, 2000). Similar to the sharing of rewards, OSS community also shares the risk of choosing a particular strategy. The reward for success or responsibility for failure of a strategy is also equally shared among the core group.

Core assumptions

The core assumptions of the OSS community include trust and loyalty (*The Apache Software Foundation*, 2001a; 2001b; Markus *et al*., 2000; Raymond, 2001). On any given module, many developers develop and implement different code segments. For this, a high level of mutual trust needs to be in place. It is also important that the core group trusts the larger community in providing solutions. Finally, shared loyalty plays an important in OSS communities (Portelli, 2000).

Table 3 summarizes the cultural aspects of OSS communities. Having examined the structure, process, and culture dimensions of the OSS communities, we show interactions among them in Figure 2. Any changes in any one dimension must be accompanied by concurrent changes in others (Hammer & Champy, 1993; Allen & Scott Morton, 1995). Within the OSS community, there exists an intricate and dynamic relationship amongst the members. For example, if community members do not value reputation (culture), the process of monitoring and sanctioning, by flaming, spamming and shunning will not work (process), and cannot be used as a basis for assigning important responsibilities (structure).

Comparing traditional and OSS environments

A comparison of organizational structure, processes and culture of OSS communities with those of traditional organizations is presented in Tables 1–3. From the comparison, it is evident that traditional organizations are rigid and unyielding. For example, traditional organizations

Table 3. A comparison of culture in OSS communities and traditional organizations

	Traditional organization	OSS community	Transition steps to hybrid-OSS
Artifacts			
Communication	Face-to-face communication	Computer-mediated communication	Encourage electronic communication among groups across locations
Location	Multiple	Multiple, global, multicultural	
Values			
Risk	Management/owner	Shared by the community	
Ownership	Management/owner	Shared by the community	
Reward	Reward structure favours owners	Reward structure is based on merit and sharing	Recognition of expertise
Motivation	Primarily financial	Altruism, reputation, ideology, financial incentives are relatively insignificant	Broaden set of motivations
Information	Information is shared on a need-to-know basis	Information is shared openly	Information sharing across groups/teams openly
Decision making	Autocratic	Almost democratic by voting	Self-management of autonomous knowledge workers
Control	Maintained by autonomous decision makers	Rules of membership, software licences, and voting procedures	
Work structures	Rigid	Flexible	
Core assumptions			
Trust	Not based on trust	Based on trust	Core developers need to work closely with one another and develop trust
Loyalty	Not based on loyalty	Shared loyalty	

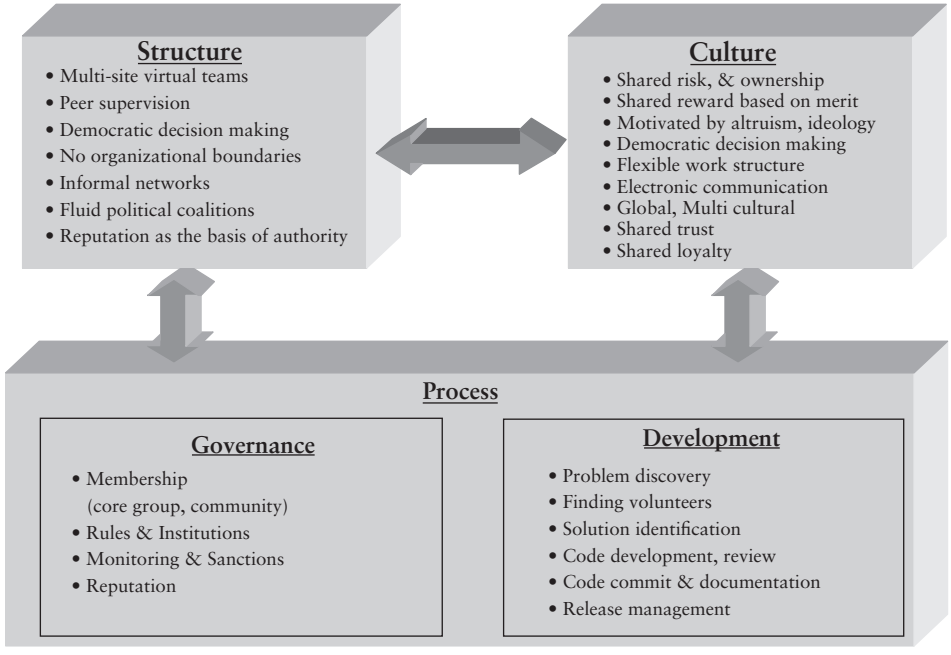

Figure 2. OSS model.

exhibit centralized control and decision-making, hierarchical governance, and constrained information flow. On the other hand, OSS communities are nimble and flexible, have shared governance, and allow free flow of information. On a continuum, the traditional organization is at one end and the OSS at the other.

We are not implying that all aspects of traditional organizations are problematic and that OSS communities have overcome all these problems. Many aspects of traditional organizations such as command and control structure, and enforcement of rules, help meet project deadlines and incorporate software engineering principles (Feller and Fitzgerald, 2002). On the other hand, OSS environments may suffer from chaos and bitter in-fighting, throwing projects off-track (Torvalds and Diamond, 2001).

Although a number of benefits can be realized by adopting the OSS model, it may not be suitable for all organizations. Organizations may have difficulty adopting the OSS model for every software development project. OSS is not a 'one-size-fits-all' framework (Feller and Fitzgerald, 2002); however, we believe that organizations can incorporate some of the salient aspects of OSS in their software development environments to attain some of its benefits. They can infuse OSS characteristics to varying degrees and move towards creating a hybrid-OSS environment, which would facilitate the development of quality software in relatively short periods of time. This will enable organizations to minimize 'time-to-market'

and remain competitive. Hence, it is imperative that traditional organizations consider hybrid-OSS environment to address some of the shortcomings in their current software development practices. Although creating these hybrid-OSS communities can be instrumental in harnessing the benefits of both traditional and OSS models, there is a great need for organizations to establish a well articulated transition mechanism in moving towards the OSS environment. To that end, we present a framework for transitioning to a hybrid-OSS environment in the next section.

FRAMEWORK FOR CREATING HYBRID-OSS COMMUNITIES

To create hybrid-OSS communities, there is the need for a systematic approach to incorporating open source practices. We propose a framework that systematically guides the creation and management of such communities within an organization. This framework contains the following three major elements: (a) community building; (b) community governance; and (c) community infrastructure, which are discussed below.

Community building

One of the preconditions for the creation of open source organizations is a large 'community of practice' with a strong, shared culture of technical professionalism (Markus *et al*., 2000). Traditional organizations may start building a 'community of practice' with a promotion of free exchange of ideas and information among their workers (Wenger & Snyder, 2000). Information sharing in traditional organizations occurs only on a 'need-to-know' basis. Free flow of information will allow workers to leverage the knowledge of others and identify potential opportunities for new innovations. To facilitate this, organizations will have to get rid of the high degree of formal structure and provide mechanisms for workers to complete tasks through informal relationships and networking.

Community governance

Shared governance

Once a 'community of practice' is in place, it must be managed in a way that is perceived as being fair and equitable by the community members (Markus *et al*., 2000; Raymond, 2001). To do so, managers must implement governance mechanisms that are transparent. Without a sense of fairness, motivation among organizational members may diminish. Managers will also have to move away from the practice of imposing a central command and control structure on the community. Community members must be allowed to work in teams and empowered to make decisions by discussion and voting. Initially, management may have a say in who becomes a part of the community. However, once the community is built, members should be able to choose what they want to do based on their skills, competence and knowledge. They

must not be forced to work in functional or divisional silos outside the community. Projects are unlikely to succeed in the absence of such strong reinforcing conditions.

Community membership management

Traditional organizations should provide mechanisms for qualified people to join the community and contribute to the project. Although, it will pose difficulties, new members must be allowed to join the community and current members leave the community. This has to occur within the parameters of both community-created and broader organizational rules and norms. Also, community members must be allowed to forge and dissolve relationships with outside entities (such as customers, suppliers, vendors, etc.) as they see fit.

Incentives and rewards

The OSS community works on meritocracy. Similarly, traditional organizations must develop a performance and measurement system, which rewards and promotes their members based on meeting both community and organizational goals and objectives. Such a system should facilitate an organizational culture in which the community as a whole is responsible for its work and gets rewarded and penalized collectively. For this system to work, community members will have to develop a high level of trust among each other.

Community infrastructure

For a hybrid-OSS environment to flourish, traditional organizations should provide the necessary tools and infrastructure for software development and project management. For example, there should be a CVS-like central repository in which the artifacts are stored and managed. Protocols for adding and retrieving artifacts from the repository will need to be well established. Before committing changes to artifacts (for example, source code), they need to be evaluated by peers for quality and generality. In addition, mechanisms for product release and documentation must be in place.

Figure 3 depicts our framework for traditional organizations to move towards a hybrid-OSS model. It is worth noting that the hybrid-OSS model, as presented in the figure, illustrates one of the many possible configurations of the hybrid-OSS environment. Organizations can draw various aspects from the traditional environment and the OSS model to create a specific hybrid-OSS structure that will meet their needs.

DISCUSSION

Developing large-scale software systems is a complex activity, which entails technical and managerial challenges. In traditional software development, considerable effort and time is

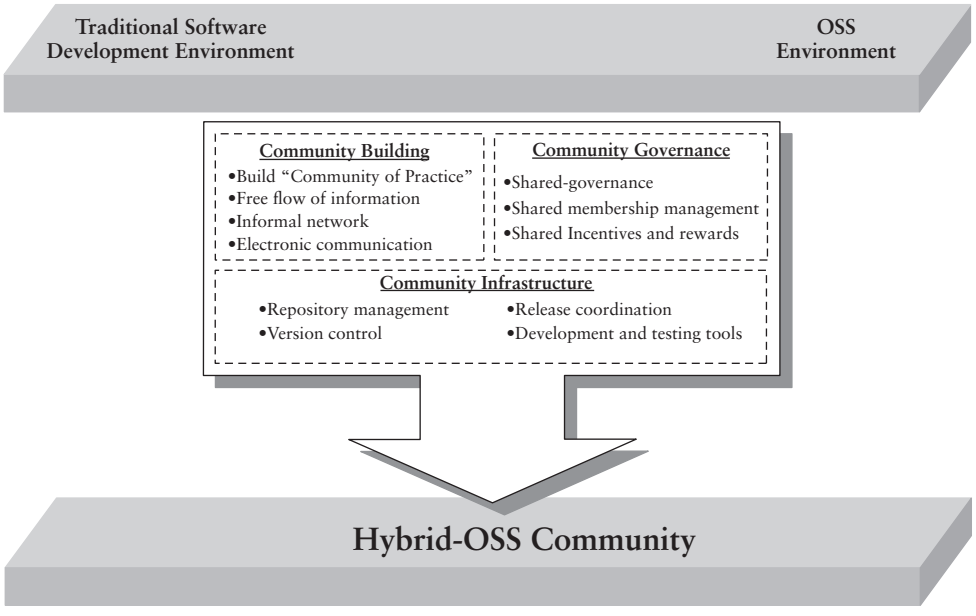

Figure 3. Framework for creating a hybrid-OSS community.

expended during the analysis and design phases to ensure that the system design incorporates important software engineering principles aimed at creating quality software. For example, modular software design minimizes coupling and improves cohesion. At the same time, strict adherence to rigid project management practices curtails creativity and forces the whole software development process to be long and drawn out resulting in cost and schedule overruns. In contrast, software development on 'internet time' requires quick completions of the project, while delivering quality software. Despite decades of research, software development is still fraught with problems because of ineffective organizational structure and processes that are in place, and the way in which software development projects are managed.

Proponents of OSS suggest that it has the potential to address several of these problems. However, OSS may not be appropriate in all cases and certainly not a panacea for all the problems plaguing the software development efforts. Although we have presented a framework that guides the creation and management of hybrid-OSS communities, the transition to such an environment should not be assumed to be a seamless process. The key to having successful hybrid-OSS environment is to identify appropriate projects and personnel. Historically, OSS development has occurred in horizontal domains (general purpose infrastructure software) where design standards exist (Feller and Fitzgerald, 2002). While a particular soft-

ware development organization may not have an opportunity to enter this horizontal market, they can develop a hybrid-OSS environment for projects that are characterized by specialized requirements in a vertical domain.

Evidence points to some leading organizations like Hewlett Packard, IBM, Intel, Sun Microsystems, etc., already having taken steps to incorporate elements of OSS into their software development environment. For those organizations that continue to rely heavily on the manufacturing model, this may be the time to think about ways of learning from the OSS model. In this regard, organizations can use our framework to understand the OSS community functions and identify opportunities (projects) in which they can use the concept of hybrid-OSS communities. In the process of doing so, they can look to reaping one or more of the following benefits: (a) reduce development time and time-to-market; (b) improve quality; (c) reduce cost; (d); gain developer loyalty; and (e) increase developer talent pool without additional head count and overhead. Indeed, the ability of organizations to move to a hybrid-OSS environment using the framework will depend, among several factors, on: (a) ability of management and workers to understand the OSS philosophy; (b) development of mutual trust between management and workers; (c) workers' perception of being involved in challenging and innovative projects; and (d) motivation of workers to participate in such projects.

CONCLUSIONS

In this information age, knowledge workers value their personal time and autonomy over greater income and advancement (Markus *et al*., 2000). Increasingly, knowledge workers are self-employed freelancers and seeking periods of less than full time of employment. With the acute shortage of qualified workers (Business 2.0, 2001), managers face the daunting task of getting projects done on time and budget. OSS community provides an ideal example of how to manage such a work-force (Markus *et al*., 2000). Unfortunately, traditional organizations have rigid structure, processes and culture that makes it difficult to provide these knowledge workers with an environment similar to the one provided in the OSS community. To create such an environment within organizational constraints, we have presented a framework.

Our research contributes to the theory and practice in several ways. Managers can use this framework to foster the creation of hybrid-OSS communities. They can also gain insights from the framework on the issues critical for the management of these communities. From the stand-point of theory, our research provides a consolidation of the literature in OSS, draws on established theories and, finally, presents a test-bed for future investigations.

There are several avenues for future research. One of the avenues is to refine the proposed framework and validate it empirically. Empirical validation can be undertaken by conducting case studies on organizations that are transitioning to an OSS-like environment. Another interesting area for study is to examine the factors that dictate why and how organizations select specific projects for hybrid-OSS development. Finally, research on effectiveness of specific strategies for transitioning to an open source-based development can guide organizational efforts in this direction.

ACKNOWLEDGEMENTS

The authors wish to thank the guest editors and anonymous reviewers for their insightful comments that helped enhance the quality of our paper.

REFERENCES

Allen, T.J. & Scott Morton, M.S. (1995) *Information Technology and the Corporations of the 1990s: Research Studies.* New York Oxford University Press, New York.

Aoki, A., Hayashi, K., Kishida, K., Nakakoji, K., Nishinaka, Y., Reeves, B. & Takashima, A. & Yamamoto, Y. (2001) A case study of the evolution of Jun: an object-oriented open source 3D multimedia library. In: *Proceedings of the 23rd International Conference on Software Engineering.*

The Apache Software Foundation. (2001a) Bylaws of the Apache Software Foundation. http://www.apache.org/foundation/bylaws.html, last accessed Sep. 13, 2001.

The Apache Software Foundation. (2001b) Apache Project Guidelines. http://dev.apache.org/guidelines.html, last accessed Sep. 13, 2001.

Asundi, J. (2001) Software engineering lessons from open source projects. In *Making Sense of the Bazaar: Proceedings of the 1st Workshop on Open Source Software Engineering*. Feller, J., Fitzgerald, B. & van der Hoek, A. (eds). http://opensource.ucc.ie/icse2001/papers.htm.

Borrell, J. (2001) Changing mankind. *Upside*, January, 68–74.

Business, 2.0 (2001) Numbers. *Business 2.0*, January 23, (2001), 111.

Connolly, P.J. (2001) Open source rules. *Infoworld*, 08.27.01/09.03/01, 64.

Cook, J.E. (2001) Open source development: an Arthurian legend. In: *Making Sense of the Bazaar: Proceedings of the 1st Workshop on Open Source Software Engineering*. Feller, J., Fitzgerald, B. & van der Hoek, A. (eds). http://opensource.ucc.ie/icse2001/papers.htm.

Deckmyn, D. (2000) Open source projects get done cheaply. *Computerworld, April*, **24**, 44.

DiBona, C., Ockman, S. & Stone, M. (1999) *Open sources: voices from the open source revolution*. O'Reilly, Sebastopol, CA.

The Economist (2001a) Business: an open and shut case. *Economist*, **359** (8221), 67.

The Economist (2001b) Survey: out in the open. *Economist*, **359** (8217), S8–S14.

Feller, J. & Fitzgerald, B. (2000) A framework analysis of the open source software development paradigm. *International Conference of Information Systems, Sydney, Australia, December 14–16.*

Feller, J. & Fitzgerald, B. (2002) *Understanding Open Source Software Development* (forthcoming). Addison-Wesley, London.

Fielding, R.Y. (1999) Shared leadership in the Apache project. *Communications of the ACM*, **42** (4), 42–43.

Galbraith, J.R. (1973) *Designing Complex Organizations*. Addison-Wesley, Reading, MA.

Hammer, M. & Champy, J. (1993) *Reengineering the Corporation: A Manifesto for Business Revolution*. Harper Business, New York.

Harvard Business Review (2000) Profiting from open source: a conversation with Rajiv Gupta. *Harvard Business Review*, September-October, Reprint F00503.

Hilson, G. (2001) Microsoft assails open source tenets. *Computing Canada*, May 18, 1.

Howard, L.W. (1998) Validating the competing values model as a representation of organization cultures. *International Journal of Organizational Analysis*, **6** (3), 231–250.

Lerner, J. & Tirole, J. (2000) The simple economics of open source. HBS Working Paper, Harvard Business School, Boston, February 2000.

Linux Documentation Project (2001) Linux Documentation Project. http://www.linuxdoc.org/, last accessed Sep. 13, 2001.

March, J.G. & Simon, H.A. (1958) *Organizations*. John Wiley, New York.

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

Masum, H. (2001) Reputation layers for open source development. In: *Making Sense of the Bazaar: Proceedings of the 1st Workshop on Open Source Software Engineering*. Feller, J., Fitzgerald, B. & van der Hoek, A. (eds). http://opensource.ucc.ie/icse2001/papers.htm.

Miles, R.E. & Snow, C.C. (1978) *Organizational Strategy, Structure, and Process*. McGraw-Hill, New York.

Mintzberg, H. (1971) *The Structure of Organizations*. Prentice Hall, Englewood Cliffs, NJ.

Moody, G. (2001) *Rebel Code: Linux and the Open Source Revolution*. Perseus Press Cambridge, MA.

Mockus, A., Fielding, R.T. & Herbsleb, J. (2000) A case study of open source software development: the Apache server. *Proceedings of the 22nd International Conference on Software Engineering, 263–279.*

Nohria, N. (1995) Note on organization structure. *Harvard Business School*, Reprint no. 9–491–083.

O'Reilly, T. (1999) *Open Sources: Voices from the Open Source Revolution.* O'Reilly & Associates, CA.

Open Source Foundation (2001) Open source software: a (new?) development methodology. www.opensource.org/halloween/halloween1.html, last accessed September 14, 2001.

Perkins, G. (1999) Culture clash and the road to world domination. *IEEE Software*, January/February, 80–84.

Plotkin, H. (1998) What (and why) you should know about open source software. *Harvard Management Update*, December, 3–4.

Portelli, B. (2000) The case for open source. *Informationweek*, December 4, 240.

Raymond, E.S. (1999) *The Cathedral and the Bazaar*. O'Reilly & Associates, Sebastopol, CA.

Raymond, E.S. (2001) Homesteading the Nooshere. http://tuxedo.org/~esr/writings/homesteading/, last accessed September 13, 2001.

Robey, D. (1991) *Designing Organizations*. 2nd edn. Irwin, Burr Ridge, IL.

Scacchi, W. (2001) Software development practices in open software development communities: a comparative case study. In: *Making Sense of the Bazaar: Proceedings of the 1st Workshop on Open Source Software Engineering*. Feller, J., Fitzgerald, B. & van der Hoek, A. (eds). http://opensource.ucc.ie/icse2001/papers.htm.

Schein, E.H. (1984) Coming to a new awareness of organizational culture. *Sloan Management Review*, **25**, 2.3–2.16.

Schein, E.H. (1996) Culture: the missing concept in organization studies. *Administrative Science Quarterly*, **41**, 229–240.

Schmidt, D.C. & Porter, A. (2001) Leveraging open source communities to improve the quality and performance of open source software. In: *Making Sense of the Bazaar: Proceedings of the 1st Workshop on Open Source Software Engineering*. Feller, J., Fitzgerald, B. & van der Hoek, A. (eds). http://opensource.ucc.ie/icse2001/papers.htm.

Sullivan, T. (2001) Open source development gaining ground. *Infoworld*, January 1, 17.

Torvalds, L. & Diamond, D. (2001) *Just for Fun: The Story of an Accidental Revolutionary*. Harper Collins, New York.

Weiss, T.R. (2000) Work set to start this month on open source lab. *Computerworld*, December 4, 70.

Weiss, T.R. (2001) Microsoft again takes aim at open source. *Computerworld*, May 7, p. 16.

Wenger, E.C. & Snyder, W.M. (2000) Communities of practice: the organizational frontier. *Harvard Business Review*, 78 (1), 139–145.

Yager, T. (2001) Open source takes hold. *Infoworld*, 08.27.01/09.03/01, 49.

Yamauchi, Y., Yokozawa, M., Shinohara, T. & Ishida, T. (2000) Collaboration with lean media: how open source software succeeds. *Proceedings of the ACM Conference on computer-supported work*, 329–338.

Biographies

Srinarayan Sharma is Assistant Professor of Management Information Systems in the School of Business Administration at Oakland University. He teaches in the areas of electronic commerce and business process redesign. His present research interests include security and privacy in electronic commerce, open source software and component-based software development, and diffusion and impact of information technology. He has published in the *Communications of the Association of Computing Machinery, Information Systems Journal* and *International Journal of Computer Applications in Technology* and numerous national conferences.

Vijayan Sugumaran is Assistant Professor of Management Information Systems in the School of Business Administration at Oakland University. He received his PhD in Information Technology from George Mason University, Fairfax, Virginia. His research interests are in the areas of intelligent agents and multi-agent systems in E-commerce, software engineering and re-use, component-based software development, knowledge-based systems, application of GIS and decision support systems. His recent publications have appeared in *Communications of the ACM, Data Base, Logistics Information Management, Automated Software Engineering, Expert Systems* and *Journal of Network and Computer Applications.* He

has presented papers at various national and international conferences.

Balaji Rajagopalan is Assistant Professor of Management Information Systems in the School of Business Administration at Oakland University. His research interests are in the areas of knowledge discovery in databases, systems development and implementation, business value of information technology and diffusion of innovations in information technology. Dr Rajagopalan's work has appeared, or is forthcoming, in *Communications of the ACM, European Journal of Operational Research, Computers and Operations Research, Information Processing and Management and Journal of Medical Systems* among others.