What makes a project open source?
Migrating from organic to synthetic communities

Siobhán O'Mahony
Harvard Business School
somahony@hbs.edu

Joel West
San Jose State University
Joel.West@sjsu.edu

January 10, 2005

Submitted to
Technology & Innovation Management Division
Academy of Management 2006

Abstract

Research on the emergence of new technical innovations emphasizes the need for a community to interpret, support, extend and diffuse them. However, little attention has been paid to the actual process of building a technical community. Recently scholars have examined one type of technical community, open source software communities, that grow organically without direction from a single sponsor. Since the commercialization of open source software, many firms have adapted this model by building synthetic open source software communities which are often conflated with their organic counter parts. Thus, we argue that the term 'open source project' no longer has shared meaning. To better distinguish between sponsored and organic open source projects, we identify three dimensions that characterize such communities: 1) intellectual property rights; 2) software development approach and 3) community governance. Drawing on ongoing field research, we find that, when creating synthetic communities, there is a central tension between controlling the community process and product and creating and sustaining the pluralism necessary to foster legitimacy and market adoption. Thus, both sponsors and contributors to synthetic communities must weigh the benefits of control against the benefits that pluralism provides. We conclude with critical challenges that synthetic communities face in their creation, growth and governance that have been neglected in previous research on open source and technical communities.

Word Count = 216

While technological innovations were once considered only in terms of their technological content, sociotechnical research has shown that technologies are shaped by human institutions that provide a context for the interpretation and use of such technologies (Bijker, Hughes & Pinch, 1987). Recent research on collective models of innovation (von Hippel and von Krogh, 2003; Van de Ven and Hargrave, 2003) and on community technical organizations (Rosenkopf, Metiu & George, 2001; Tushman and Rosenkopf, 1998) has emphasized the role of technical communities in not only interpretation and use of such technologies, but in their creation and further development. Technical communities provide a forum to coordinate the resources of firms and individuals in order to develop new technologies (Rosenkopf et al, 2001).

In this paper we consider how firms create and sustain such communities by examining a specific type of community, those that support open source software development. We consider the differences between autonomous, sponsored and hybrid forms of open source projects. To explain the differences between these types, we identify three dimensions of open source projects: intellectual property rights, development approach, and model of community governance. We use these dimensions to classify communities grown organically by individuals as well as those grown synthetically by sponsoring firms. In doing so, we isolate a central tension of synthetic technical communities: asserting control to direct the technology to ally with the founders' goals and relinquishing control to attract others to support the project's broader goals.

From this analysis, we offer a model explaining the synthetic growth of technical communities. We conclude with implications for technical communities and suggestions for integrating future research on technical communities in general with research on open source software communities.

Innovation, Technical Communities and Firms

The Importance of Technical Communities to Innovation

Path breaking innovation cannot occur without a community or institutional field to interpret, support, extend and diffuse it (Van de Ven and Hargrave, 2003; Schoonhoven and Romanelli, 2001; Hargadon and Douglas, 2001; Hunt and Aldrich, 1998; Christensen and Rosenbloom, 1995; Rosenkopf and Tushman, 1994; Tushman and Rosenkopf, 1992; Anderson and Tushman, 1990; Van de Ven and Garud, 1989). Technical communities also provide a vehicle for the exchange of technical, possibly proprietary, information that fosters cumulative innovation (Saxenian, 1994; von Hippel, 1987; 1988; Allen, 1983). Thus, when explaining how new technical innovations emerge, theorists now extend their unit of analysis beyond innovating firms to the communities and eco-systems necessary to support the adoption and dissemination of their work, particularly for technologies that depend upon a common platform (Garud, Jain and Kumaraswamy, 2002; Hargadon and Douglas, 2001).

First, new technologies must invoke existing understandings of the established institutional environment in order to be assimilated and adopted (Hargadon and Douglas, 2001) and this requires collective action (Van de Ven and Hargrave, 2003). “Purely novel actions and ideas cannot register because no established logics exist to describe them” (Hargadon and Douglas, 2001: 478). If an invention cannot be understood, it will be difficult to create market demand for it and diffusion will be thus hampered. The construction of meaning must take place at the institutional field level and be shared by many actors in order for a particular technology to move from invention to innovation.

Second, the degree to which collective action is required depends on features of the technology itself. The greater the complexity, the greater the number of sub-systems or

complementary components that are possible, the more linking mechanisms or interface technologies are required, the greater the likelihood that the construction of a new technology trajectory or dominant design will be a social and political process (Tushman and Rosenkopf, 1992). In other words, modular open systems with standard interfaces (or "design rules") that enable other industry players to create components that can work together within a larger common architecture (Baldwin and Clark, 2000) invite a greater number of stakeholders. This may help explain why the community development model has become popular in the software industry. Staudenmayer, Tripsas and Tucci call these systems "development webs" when no single firm or institution controls the architecture (2004). Instead, networks of organizations with multiple diverse constituencies help negotiate competing parameters to define design rules (Tushman and Rosenkopf, 1992; Rosenkopf and Tushman, 1994).

A long line of research on the emergence of new technical innovations shows that, in the face of competing design options, new technologies rarely win on the merits of their features alone (Tushman and Rosenkopf, 1998; Tripsas, 1997; Van de Ven and Garud, 1994; Anderson and Tushman, 1990; Tushman and Anderson, 1986). Coalitions shape the articulation of competing technical alternatives and settlement on the parameters of new innovations (Tushman & Murmann, 1998). However, we know very little about how firms actually manage their relations within these coalitions. Furthermore, coalitions may be composed not just of firms, but of community technical organizations (CTOs) (Rosenkopf and Tushman, 1998) and individuals, creating confusion as to the relevant unit of analysis (Rosenkopf et al, 2001).

Firms and Technical Communities

There are several reasons why firms might collaborate with technical communities. They can glean information on the possibility of alliance formation and identify opportunities for future

collaboration (Rosenkopf et al, 2001). They can also share risk on developing innovative projects that might affect the trajectory of technological innovation in an industry or the emergence of de facto standards (Tushman and Rosenkopf, 1992; Rosenkopf and Tushman, 1994; 1998). This includes an implicit, but rarely stated goal: to control or influence the standards process in a direction that is positive for the firm (Rosenkopf et al, 2001; Van de Ven and Garud, 1998; Pfeffer, 1981). For a firm with more control over the direction of their industry (and by extension, less uncertainty) is more likely to succeed in their endeavors (Pfeffer and Salancik, 1978).

Research has shown that technical communities can support the development of individual firm representatives and that individual collegiality with employees from other firms can improve inter-firm relationships (Rosenkopf et al, 2001). Technical communities also offer individuals leadership opportunities (Fleming and Waguespack, 2004; O'Mahony and Ferraro, 2004); enhance their technical credibility and signal their availability to prospective employers (Hars and Ou, 2002). For example, participation in a specific public community (such as standardizing the http protocol as a member of the Internet Engineering Task Force) allows participants to both develop and advertise domain specific knowledge.

Despite the fact that there is growing recognition of the type of collective action necessary to produce new technical innovations and increased interest in relations among firms and technical communities, micro-interactional research on community building processes is scarce. The micro-level perspective on the social and political processes necessary to build and sustain technical communities needs to be unpacked (Van de Ven, 1993; Van de Ven and Hargrave, 2003). How do communities, ecologies or development webs evolve? What roles do individuals play? What roles do firms play? As community building becomes a corporate sanctioned activity

as opposed to an emergent grass roots activity, how do community building practices change? In what ways do the dynamics of technical communities differ from traditional standard setting organizations? Furthermore, if the concept of technical community crosses levels of analysis at the individual and firm levels, technical communities might be an important meso level construct that link individual and voluntary collective action to innovation.

Furthermore, participation in technical communities is not without its costs (Millen, Fontaine and Muller, 2002; Rosenkopf et al, 2001). In addition to absorbing costs for the work time of employees, travel, membership fees and other sponsorship costs, firms "bear the risk that they will lose proprietary information to competitors through the interactions that occur in CTOs" (Rosenkopf et al 2001: 749). It is easy to imagine how this risk might be exacerbated when collaborating with technical communities such as the open source community which values the development of non-proprietary software. Yet, this has not stopped firms from either participating in open source software projects or starting their own. Examining firm-community collaboration in open source software provides a new opportunity to extend prior research on technical communities to a new organizational context. It requires however, revisiting the definition of open source project.

The Many Meanings of Open Source

Since the term "open source" was coined in 1998, "open source" software development projects have attracted a wealth of scholarly attention. While the term "open source" initially represented a distinct class of software licenses, in practice, this term is often used to conflate what we argue are three distinct dimensions that shape the socio-technical structure of communities: 1) intellectual property policy, 2) development approach, and 3) a community governance model (Table 1).

First, the Open Source Initiative (OSI) certifies specific software licenses that meet the definition of "open source". Such licenses must provide rights to use, modify, and distribute source code. Second, the open source development process is often described as volunteers loosely organized into virtual teams, openly collaborating with a common set of software tools and Internet-enabled communication process to produce an information good. Third, open source is also used to refer to a community governance model where control of a project, its resources and outputs are shared among project participants, and crucial project decisions are resolved using a mutually agreed and predictable system of decision rights.

In practice, projects and firms have experimented with variants of each of these dimensions. Of the three, experimentation is most limited for intellectual property policies, which are constrained by OSI guidelines.[1] Even within these constraints, the number of approved "open source" licenses has increased as firms continually create new licenses in the hope of earning OSI certification.

Experimentation for the other two dimensions have no such constraints. While projects frequently copy each other's best practices, there is no comparable shared definition (or legal or normative constraint) for the development and community dimensions of an open source project. Thus, today the term "open source project" does not have a clearly defined stable and shared meaning comparable to "open source license."

Firms have experimented with organizing projects that contain some but not all of the dimensions of open source project. For example, Microsoft's "shared source" initiative offers a

---

[1] However, more recently Parker & Van Alstyne (2005) have proposed modifications to open source and free software licenses to provide temporary monopolies and thus additional incentives for innovation investment.

subset of open source intellectual property rights to defined sub-groups of customers (West & Dedrick, 2001). Sun Microsystems has also created projects with a community development process, the Java Community Process, which has a select subset of intellectual property rights (West, 2003). Meanwhile, many firms are managing "gated communities" that use open source development processes for a defined population without sharing either intellectual property or governance (Shah, 2004).

Firms have also experimented with different forms of community formation and governance. In particular, the control of an open source community seems closely related both to the origins of the community and the definition of ongoing membership (Table 2). Of course, many projects remain tightly controlled by sponsors, in which there is no concept of membership and non-sponsors have no formal voice in the direction of the project or community.

In the remainder of the paper, we contrast two fundamentally different forms of such communities — those that grow organically as a self-starting, self-sustaining technical community and those that are grown synthetically by external sponsors (typically firms) to support the sponsors' ends. Between pure proprietary and community managed open source projects, we find many approaches to hybrid forms. In examining what constitutes a hybrid, we highlight the competing goals of voluntary collective action within technical communities: asserting control to direct the technology to ally with the founders' goals and relinquishing control to attract others to support the project's broader goals.

Organic Open Source Projects

The bulk of open source research has focused on organic open source software projects founded by individual volunteers supported by the Internet independent of their employment context. Familiar examples included Linus Torvalds' founding of the Linux operating system

project, Miguel D'Icaza's initiation of the GNOME desktop environment project, and the adaptation of the NCSA web server by eight developers to form Apache. A project is community managed if it is composed of individuals who do not share a common employer and if employment relations do not guide project relations (O'Mahony, 2005). A firm may sponsor contributors, but firms typically cannot become members. Furthermore, sponsored contributors must earn and sustain their role on the project under the same terms as volunteers. Thus, for firms to collaborate with a community managed project, collaboration is enacted through sponsored individuals (O'Mahony, 2002).

Motivation

Volunteer contributors to community managed projects are motivated by a complex set of personal and career concerns. Lerner and Tirole (2002) argue that contributors to community projects participate in order to improve the visibility of their skills in the open labor market. A more recent survey of SourceForge developers shows that most contributors to community projects are interested in building their skills and solving technical problems and that career benefits are of secondary concern (Lakhani and Wolf, 2003).

Hertel and colleagues surveyed 141 Linux contributors and found that participants were motivated by their identification as a Linux developer, by pragmatic motives to improve the software for their own use, and by their ability to commit their time (2003). Volunteer contributors write code in order to solve technical problems that matter to them (Hertel et al, 2003; von Hippel, 2001; Raymond, 1999). However, the motivations of sponsored contributors are likely to be even more complex as they are affected by their sponsor's goals.

Intellectual Property Rights

A project's software license is chosen by its founding individuals and may also affect the ability of a project to attract developers (Stewart et al, 2005; Stallman, 1999). However, most community managed open source projects use a few well known licenses that have a long history of acceptance in the larger software community. Lerner and Tirole's study of 38,610 open source projects found that most used the GNU General Public License written by Richard Stallman in 1985 (2003). This license allows free use and modification of source code but requires derivative works that are redistributed to be licensed under the terms. This requirement of reciprocity effectively prohibits proprietary appropriation. This helps assure contributors who might otherwise worry that their work might be distributed under proprietary conditions (Stallman, 1999).

However, there are many licenses that meet the definition of open source that do not require reciprocity and provide developers with greater flexibility in deciding how to treat their own software (Lerner and Tirole, 2002). The Open Source Definition, created in 1997 requires that a license need not insist that derived works distributed on the same medium must also be licensed as open source as well. One well known example is the Berkeley Software Distribution license which shares some similarities with the GNU GPL but relaxes the requirement to relicense distributed derivations under the same terms (Lerner and Tirole, 2002). Both of these licenses have been in use for a long time: their commercial limitations and affordances are well known.

Development

Community founded projects grow organically: the community and code scale in parallel. Founders often make the critical design and implementation decisions and the code slowly scales as other developers working in tangential areas join the community. Incumbent project members

may "test" a newcomer's ability to contribute to a project by evaluating their ability to articulate a contribution to the code base. Potential members create "joining scripts" specifying a unique contribution to become an accepted contributing member of the project (von Krogh, et al, 2003). In this manner, the complexity of the code grows as new developers join.

As a result, new community members are incorporated early in the design phase. Building community in parallel with the code base allows early entrants to lay their fingerprints directly on the architecture of the project, enhancing the robustness of the architecture itself; fostering shared formats, processes, and norms; and increasing individual commitment to the project. This can create confusion and ambiguity in the short term, but enable a more robust project architecture and greater individual commitment in the long term. For example, very early in the development of Linux, Torvalds relied on others to design and implement crucial networking libraries, developing the "lieutenant" system of senior technical leaders that remains today (Moody, 2001). At the same time, this process can also be marked by long periods of ambiguity over project direction and viability and take time to develop critical mass.

A modular project architecture may also help a project scale in a decentralized fashion. Baldwin and Clark (2003) show that projects that are more modular are better positioned to recruit new contributors. A greater number of modules can offer more opportunities for recognition, which can further enhance contributors' motivations (Baldwin and Clark, 2003). Early participation in the design phase can increase individual commitment to the project.

However, community managed projects appear to sustain extremely uneven rates of participation. Several studies have found, upon close examination of code contributions, that most community managed projects rely upon a small distinct core group and receive contributions from a much larger group of legitimate peripheral participants. For example, on the

Apache project, 4% of 400 programmers in Apache contributed 88% of the code (Mockus et al, 2002) and on the GNOME project, 22% of 301 programmers contributed 80% of the code (Koch et al, 2002). Similar distributions have been found for the Linux project (Hertel et al, 2003) and in surveys of large scale multi-project databases (Ghosh and Prakash, 2000; Dempsey et al, 2002; Xu et al, 2005). How such uneven patterns of contributions affect a project's social and technical has not been explored.

In their study of projects hosted by the Sourceforge website, Xu and colleagues found that the vast majority of developers only contributed to one project while the top 100 developers contributed to 1886 different projects: a ratio of 1 contributor to 19 projects. Thus, the pattern of core and periphery participants may apply not only within projects, but across the entire open source ecology – suggesting the need for more cross-project studies at the ecology or institutional field level.

Governance

Community managed open source projects are often portrayed as self-organizing entities. In practice, projects that are large, mature and commercially distributed have developed informal and formal governance mechanisms that enable representation in commercial and legal settings. When mature projects such as Apache, GNOME, and Debian achieved global recognition for their work, the number of interested contributors outpaced the projects' ability to welcome them in the informal capacity that marked their organic growth. For example, under a deluge of applicants, Debian temporarily closed its doors to new members to design a membership process (O'Mahony and Ferraro, 2004). Apache, Debian, and GNOME responded to these challenges by designing formal membership criteria and representation mechanisms to manage an onslaught of new recruits and new challenges from commercial entities.

Attention from investors, analysts, the media, and potential commercial collaborators brought new types of list subscribers, inquiries and contributors. Project members found some of their informal decision mechanisms inadequate for making formal commitments, guiding the project's direction, and legally representing the project (O'Mahony, 2002). Thus, many projects created non-profit foundations to hold project assets and help oversee project direction but not day-to-day technical decision-making (O'Mahony, 2002; 2005). These foundations often offer roles to other organizations that support or extend a project's work. Non-profit organizations such as the Open Source Development Lab, firms developing complementary products, or other community projects with which they collaborate may participate in advisory councils, committees or boards.

What may be unique to organically founded open source projects, is that the governance mechanisms designed, at their roots, reflected longstanding project norms. Projects like Apache, Debian, and GNOME were focused on correcting current problems: disseminating and institutionalizing established project norms to a larger and more diverse population and thereby helping to sustain the project's original direction. Governance was not imposed, but emerged out of shared perceived need. Furthermore, by collectively creating governance mechanisms in a project's later life, newer project entrants who did not participate in the project's initial technical architecture found an additional way to participate in a meaningful way, thereby amplifying individual commitment to the project.

Clearly, the organic approach offers many advantages in terms of recruiting individual volunteers seeking distinct technical challenges and encouraging ownership and commitment to the project. The main disadvantage of the organic approach is that startup costs — both organizational and technical — are fully born by the community. In its purest form, volunteers establish structure, collaboration technologies and define roles for participation all while trying

to build a working 1.0 system. Not surprisingly, only a small percent of open source projects founded on the SourceForge.net website, the largest repository of open source software, have built a complete system or created a thriving online community (Krishnamurthy, 2002; Healy & Schussman, 2003). Only 75% of 46,356 projects hosted on SourceForge had any software in their code repository and 95% of the projects had 5 or fewer registered contributors (Healy & Schussman, 2003).

Granted, the Sourceforge hosting site may only serve as an incubator for fledgling ideas that might become projects. In such a forum, starts and failures are, naturally, more visible. Not every project will share the same potential to scale as projects like Linux. Linux, like other mature, successful organic projects, had a longer gestation period: it was founded at least eight years before the emergence of a sizable market for it. However, with an increase in the number of projects founded, there is likely greater competition for donated labor. Thus, many organic open source projects do not seem to make it past the incubation stage for some combination of these reasons. Furthemore, the continued reference to the same handful of successful projects (notably Linux and Apache) suggests that there is tremendous difficulty building projects this way.

Sponsored Open Source Projects

Since Netscape publicly released the source code for its Internet browser, Mozilla, in 1998[2], an increasing number of public and private sponsors have released code created under proprietary conditions in order to grow an external community to improve the future code base. While Netscape was widely credited as the first company to explicitly pursue an open software

[2] For more information, please see Netscape's press announcement located at: http://wp.netscape.com/newsref/pr/newsrelease577.html. This browser is now more commonly known as Firefox located at: www.mozilla.org.

development strategy by building an external supporting community, little research has examined the execution or efficacy of Netscape's strategy. However, neither a lack of data nor Netscape's demise as a firm has stopped other companies from sponsoring open source development projects.

Table 3 lists examples of major sponsored open source projects founded since 1998.[3] Sponsors may intend for the community to either supplant or replace prior proprietary development efforts, but most often they will use community developed code as a platform for proprietary development work. Sponsors can play different roles in the project, both in its creation and its ongoing support (Table 4).

At the foundation of every open source project lies shared intellectual property. However, sponsors face very different challenges from community-founded open source projects as they must manage the boundary between intellectual property that is shared among voluntary contributors and competing firms while crafting their own differentiated offerings. If we are to develop a richer and more accurate understanding of how firms use intellectual property as a source of innovation and competitive advantage, than we need to examine why and how sponsors develop open source projects.

Motivation

When a sponsor initiates an open source project, they open project development not just to a community of individual volunteers, but to a community of potential firm collaborators and rivals: engaging in what Chesbrough (2003) terms "open innovation." We can suggest two reasons why sponsors would initiate an open source project.

---

3 We omit projects that are subprojects of established open source projects, and thus accept the IP, development processes and community governance rules of the established projects. An example of such a project would be Beehive, founded in 2004 by BEA under the rules of the Apache Software Foundation.

First, sponsors may want to create a larger market for the project's software or reduce a competitor's market share by building a robust development ecology (West, 2003). A larger pool of innovators can supplement a firm's internal resources (von Hippel and von Krogh, 2003; von Hippel, 2001) and enhance complementary innovations by third parties (West and Gallagher, 2004). An open source project may also increase public awareness, accelerate distribution, and reduce the costs of marketing.

Secondly, firms may have software that is potentially valuable to an external community, but has failed the criteria for further in-house development (West and Gallagher, 2004). A firm might want to divest itself of such a commercial product so that it can redeploy its resources elsewhere (West, 2003). Furthermore, if a community emerges to maintain the code, customers of the technology may still be able to receive support without draining the resources of the firm.

Alternatively, a firm may not have been able to market a technology that would otherwise be "sitting on the shelf" (Chesbrough, 2003). By sponsoring an open source project, a firm can assess the market's receptivity and identify potential 'false negative' decisions (Chesbrough, 2003). For example, if the technology is more successful than the firm had estimated, the firm may reevaluate its business decision and engage in further development. In either case, the sponsor may create goodwill (and perhaps add value to related products) by releasing code to an external community, while minimizing its own ongoing costs and responsibilities.

Research on the motivations of volunteer open source contributors has focused primarily on organic projects. However, the motivation of external contributors to sponsored projects is likely to be more complex primarily because the software will initially be sponsor owned. Although ownership and governance of the code may be transferred to an independent body at a later

stage, newcomers to a sponsored project enter an established social technical system with norms and rules created under hierarchical employment relationships.

However, sponsorship can provide resources, legitimacy and technical capabilities that improves the odds of project success relative to organically founded projects. By providing a solid technical foundation for future innovation, a sponsor reduces the risk that contributors will find the project defunct before its software is usable. For example, Stewart and colleagues (2005) found that projects sponsored by firms or non-profit organizations were more likely to become more popular over time.

Intellectual Property Rights

Nonetheless, contributors to sponsored projects may be more likely to worry about their future legal rights to use, modify and distribute the code they help develop than they would on community managed projects. As mentioned earlier, most community managed projects are licensed under the GPL, which ensures that the future stream of development work will be publicly available. Use of the GPL on sponsored projects may encourage greater dissemination of a unified distribution, but discourage provision of complementary solutions.

Firms will be less likely to invest in complementary solutions if they can not be made proprietary. Thus, sponsored projects tend to use a wider range of open source licenses, most of which lack the reciprocal sharing (aka "viral") provisions of the GPL.[4] There are over 54 approved open source licenses and over half of them are either product or firm specific (OSI, 2004). Firm or product specific licenses do not share a long history within the software community and may introduce additional complexity. The average individual volunteer

[4] For example, popular non-viral license such as the Apache, BSD, or Mozilla licenses allow proprietary extensions and are familiar to the open source community.

contributor may not be well versed as to the nuances of attribution, modification and distribution terms of product or firm specific licenses. Firm specific licenses may also reinforce the perception of sponsor control and reduce the legitimacy of efforts to create shared intellectual property.

A final concern for sponsors wishing to create an external community is whether to divest themselves of the ownership of the code. IBM did this by assigning their copyrights and trademarks to the Eclipse software to a non-profit foundation to hold the code in trust. However, as Table 5 shows, many sponsors of open source projects do not take this route and choose to retain ownership of the code. We would expect that sponsors that create an independently governed body to hold the future stream of intellectual property derived from the sponsor's code base will be more likely to attract multilateral contributors.

Development

Like community managed projects, sponsored projects also require a modular decentralized structure where well defined user and account management rights control different sections of the technical architecture (Baldwin and Clark, 2003) as the opportunity structure for external participants is affected by the discrete technical challenges available. Unlike organic open source projects, sponsors of open source projects also invest substantial resources to transition to a community of external contributors, particularly if they are opening up their development process to receive contributions of code, or bug reports from individuals or other firms.

However, not every sponsor of an open source project opens their code to receive contributions from outside parties. An alternative to an open development model is closed but transparent development. That is, if the sponsor retains full control over software development, the project remains a closed development effort, but with greater transparency to outsiders such

as those that supply complements. This is particularly true if (to use the terminology of West 2003), the sponsor is only "opening parts," disclosing a subset of the full range technologies that most users would need to build a complete working system. An example of this is the Darwin project, where Apple released part of its OS X as open source but kept key components proprietary (West, 2003).

The result may resemble proprietary development conducted within a fish bowl, where outsiders observe but only participate at the margins of the sponsor's development efforts. The source code may be partially shared, and development is transparent, but individuals are not able to directly contribute to development and the project is thus not community managed. Sponsors transitioning to a community model may temporarily pursue the transparent but closed development model in order to involve and educate an expanded community for a greater future role. For example, the Open Source Application Foundation is using this approach to garner interest in the Chandler personal information manager until their architecture is robust enough to support community development. The Mozilla browser and Jikes compiler pursued similar approaches early in their lifecycle (West & Gallagher, 2004).

A related but distinct example of ongoing sponsor control over development can be found in "dual license" open source projects such as MySQL, in which the sponsor gains revenues from selling commercial licenses to firms but also licenses their software under the GPL on terms acceptable to individuals and non-profit users (Välimäki, 2003). As an example, half of the 5.8 million in revenues earned by the Swedish company MySQL AB in 2004, came from commercial licenses of GPL licensed software (Red Herring, 2004). We would expect that sponsors of open source projects using dual license models would be more likely to rely upon closed but transparent development models.

Governance

Our preliminary analysis of data on sponsored projects found that sponsors varied in the degree to which they created structures to support multilateral decision-making. Table 5 shows three types of governance structures identified thus far: sponsor controlled, firm member dominated, and community managed. Sponsors that do not choose to divest control of their project remain, in effect, sponsor controlled. If sponsors do not create a non-profit foundation to hold the project's assets, they will not have a vehicle to offer membership to either individuals or firms. Sponsors that created independent governance structures allow both individuals and firms to participate[5]. Firm member dominated models such as the Free Standards Group which supports the Linux Standards Base and the Eclipse Foundation offer membership to both individuals and firms, but governance positions are dominated by firms.

A community managed form of governance is, operationally, closest to the organic forms of governance created by early open source projects. For a synthetically created project to become community managed, an active sponsor must divest itself as a dominant supporter of the project and recruit a vibrant external community to share ongoing control and responsibility for developing a complex evolving system. Establishing a sustainable independent governance model will require time and during the transition, sponsors may retain a majority role. However, we would expect that sponsors that articulate how the process of divestment and subsequent sharing of project responsibilities and rights will begin will be more highly received by extant technical communities than those that do not carefully plan such a transition.

---

5 Foundations that support collaboration among individuals and firms charge differentiated fees for different classes of membership and may waive individual fees in exchange for their donated labor.

Our review of research on community founded and sponsored open source projects shows that researchers know much more about the former than the latter. Our preliminary analysis has uncovered some hybrid models that fall between proprietary and open source models. These hybrids differ in the level of access, transparency and plurality provided. Sponsors have been more successful in providing transparency in their development process and access to their source code than they have in establishing models with open development processes or shared governance.

Ultimately, a successful sponsored community project would utilize a broad base of contributors, be vendor neutral and self-sustaining. If the Mozilla project is any indication, some projects may transition through different models before finally creating a community managed open source project. While the Mozilla project may have started as a sneak preview project in 1998, it is now collectively owned and managed by an independent non-profit foundation and, in all respects, community managed. Without conclusive data on these models, we conclude that sponsors are still experimenting with adapting elements of the organic open source model to meet hybrid proprietary-community needs.

In transitioning from a sponsor managed to a community managed open source project, we identified four roles enacted by sponsors: 1) *providing code* for a working system; 2) *providing resources,* such as hardware, web hosting, marketing support or ongoing programmer time; 3) *transferring knowledge* regarding the code's history, design and structure; and 4) *community leadership* in allocating decision and legal rights and responsibilities in the transition to a shared governance structure. While a formal sponsor can reduce ambiguity and provide a solid technical foundation for a community managed project, our preliminary data suggest that crafting clear

contributor rights and shared governance will be critical to attracting external contributors. We explore this tension more fully in the next section.

Comparing Sponsored and Organic Open Source Projects

We have highlighted some differences between the organic (community-led) and synthetic (sponsored) models that will affect their ability to recruit developers, coordinate, and develop software. Projects that are spun-out from corporate sponsors may have several advantages. Their architecture may be more clearly defined, their viability more clearly established, and they may have received commercial and/or community recognition that can help attract talented developers. Sponsored projects have better assurance of the support needed to develop and maintain a "commercial face" to market the project and promote it at conferences and tradeshows. However, these projects lack several dimensions that have previously been associated with 'open source projects'.

Projects spun out after extensive internal development will not have had community participation in the design of the technical architecture. Thus, gaining community participation in the design of project governance will be critical to establishing legitimacy and managing the appropriate level of modularity will be even more important to attracting talented developers seeking challenging work. Contributors will not be able to stake out sections of the project to which they can uniquely contribute in the presence of prior "owners" to the code. Implications from prior work on organic projects (Hertel et al, 2003; Lakhani and Wolf, 2005) suggest that if contributors do not see how they can "make their mark," they may be less inclined to contribute to a sponsored project.

Potential contributors may react negatively to spinouts that aim to transfer responsibility to maintain the code to a community as opposed to creating a collaborative partnership. Thus, divestment inspired approaches may be more risky than approaches motivated to grow a particular market. If the sponsor does too much pre-emptive governance design, potential contributors may question the sponsor's motives and be reluctant to become involved. On the other hand if the boundary between commercial and community ownership and control is not secured, potential contributors may be less motivated to contribute. Potential contributors will be quick to note if pluralism is encouraged, but the governance structure allows a dominant to emerge.

Introducing a community to an established spinout project post hoc also raises new technical, relational, and legal challenges that do not apply to community-founded projects. With a spinout project, the infant community is presented with a large complex system that may be hard to decipher at macro and micro levels. The code will be even more difficult to learn if it was developed by a single author or a small team, in which the overall architecture and design goals remain unspoken tacit knowledge. The community may have trouble developing a sense of ownership as newcomers do not benefit from the (often crucial) intrinsic motivation that comes from building a complete system from the ground up. Thus, the sponsor and potential community members may have different visions, goals and priorities. Ultimately, to generate participation from an external community, the goals of the project must be perceived by potential contributors to extend beyond the goals of the firm.

Our research suggests that firms experimenting with the open source model are more likely to do so by combining proprietary and open source processes to create new hybrid forms, but the success of such forms is not yet proven. As elements of organically founded open source projects

are adapted by firms, these hybrid forms will become more common, and examining the range of these alternatives and their eventual outcomes is important to understanding not just the commercialization of open source software, but the evolution of collective models of innovation.

Sponsored open source projects share many of the same growth issues as community managed projects, such as building a collaboration infrastructure, designing governance mechanisms and making key decisions about licensing and external relationships. However, ongoing relations between the sponsor and members new to the emerging community introduce different questions of ownership, decision rights, and control. West (2003) asserts that firms making technology investments face an "essential tension" between appropriating returns from their investment in the community and providing incentives to recruit contributors, adopters and complementors.

Factors Affecting the Balancing of Community and Sponsor Interests.

To the degree that sponsors assert unilateral ongoing technical leadership and control, sponsored projects may have difficulty recruiting external contributors. Independent volunteers and sponsored contributors will be reluctant to donate resources to projects where the governance structure is unclear or where access, use and distribution rights are not universal. At the same time, sponsors must walk a fine line between asserting preferential decision rights and losing all influence over the project.

A well functioning governance model should enable the sponsor to influence the project's evolution through coalition building and pluralistic support. Thus, a sponsor's desire to establish licensing and governance terms favorable to its own interests must weigh against providing incentives for participants to join and contribute to the community. We predict three factors that will affect the sponsor's approach to community building: 1) degree of interdependence with

complementary work; 2) length of software development cycle; and 3) degree of market share for the project.

We expect that firms that are able to decouple complementary work from a community project will be more willing to share development and governance of the project with an external community. The importance of a firm's complementary work to their revenue stream may also affect the ability to which a firm is willing to share control — simply because the more the sponsor's business model depends on the software, the more the firm will benefit from greater control over development. Thus, we would expect sponsors building tightly coupled complements to be less likely to relinquish control of highly interdependent projects.

Second, we would expect sponsors with longer product development cycles to be more likely to relinquish control, while those with shorter product development cycles will be more likely to assert greater sponsor control. As IBM's top Linux engineering manager noted, "The open source process doesn't really work against fixed deadlines" ("Interview with Dan Frey," 2003). Thus, we would expect firms with shorter development cycles to be unlikely to depend on external resources to meet urgent deadlines. Third, if the software is in early stages or does not yet have a large market share, then the sponsor may focus on attracting new contributors, developers and users and be willing to relinquish more control. For example, sponsors interested in testing prototype ideas may be more amenable to a participatory approach to community development than sponsors with software that already has well established markets.

Integrating Sponsor and Community Resources

We have identified an implicit trade-off between the degree of control a sponsor can exert and their ability to create a successful self sustaining open source community. We propose a relationship mediated by the opportunity structure the sponsor provides (Figure 1). When

sponsors initiate a community managed project, they typically provide code, supporting hardware, human resources, and marketing resources to the project. We propose that, in addition to the nature of the technical challenge, the degree of control the sponsor releases affects the opportunity structure for external participation.

The opportunity structure, will in turn, affect the project's ability to attract resources from volunteers and other organizations. Greater community resources can increase the project's total resources and thus enhance the project's efficacy and sustainability (Figure 1). Asserting unilateral control; limiting external access to the development process; restricting rights to the community's output or foreclosing (whether explicitly or implicitly) the rights of external volunteers to share in community governance may narrow the opportunity for external contributors to participate and decrease the probability of others offering material support to the project.

The efficacy or health of a collaborative technical community is often measured by quantity of output — either frequency of releases or frequency of updates (Lee and Cole, 2003; Simcoe, 2004, Stewart et al, 2005). Other measures could include measures of output quality (bug identification and resolution rate) or measures of the health of the community, such as internal consensus or breadth and composition of participation. How volunteer experts and sponsored contributors from other organizations are represented among developers, implementors, complementors and customers will be an indicator of plurality.

What is unknown, is how a model of shared control will affect the firm's ability to capture value. Sponsors have dual objectives — to make the community successful and to make sure that the community's success supports the sponsor's goals. We would expect that sponsors seeking cost reductions will be interested in creating an effective, self-sustaining project that is successful

on its own terms with few ongoing resources from the sponsor. Sponsors with more strategic goals will expect the community to facilitate the sale of related complements (such as development tools and web servers) and services. These types of sponsors are likely to be more concerned with measures of the software's adoption rate, market share, and brand reputation. Non-commercial sponsors (such as non-profit foundations or government agencies) will be equally concerned with awareness and adoption, but more focused on the growth of digital resources to support the public good.

Conclusions

Our study highlights the general lack of empirical research on sponsored open source projects (Stewart et al, 2005 a recent exception). We have argued that the multiple meanings of 'open source project' have reduced the shared meaning once associated with the term. By analyzing how community and sponsored projects differ with regards to their intellectual property; development models; and governance structure, we have clarified some of the elements that constitute the emerging hybrid models that fall between proprietary software projects and organic community managed open source projects. We have proposed how these differences might affect the respective growth trajectories of community and sponsored projects and the factors that might lead firms to create one type of sponsored project versus another. We conclude by suggesting avenues for future research on sponsored open source projects that would contribute to the broader literature on technical community organizations.

Integrating Open Source and Technical Community Research

We believe that research on sponsored open source projects can reveal much about conditions that foster cooperation and competition in technical communities writ large, but its deeper relevance to the extant literature on technical communities is yet to be realized. Currently,

there are two major streams of research on collaborative technical communities: research on the standards-oriented community technical organizations (Rosenkopf et al, 2001; Rosenkopf and Tushman, 1998; Fleming and Waguespack, 2004, Simcoe, 2004) and parallel work on open source communities (Lakhani and Wolf 2003; von Krogh et al, 2003; Franke & von Hippel, 2003; Franke & Shah 2003; von Hippel and von Krogh, 2003; Lee and Cole, 2003; O'Mahony 2003, 2004, 2005; Kogut and Metiu, 2001; von Hippel, 2001). Both streams address similar phenomena, and yet have essential differences. Both consider the challenge of motivating and organizing voluntary collective action to produce a public or collectively shared good. The contributions of one firm can aid another firm who may be a direct rival. If access, modification and distribution rights are universally shared, the benefits of contribution will not be limited to participants, whether it's 3COM (in co-sponsoring the Ethernet standard) or IBM (in providing major funding for Linux development) but to all hoping to use the technology.

These two streams differ in the organizing artifact and mission of their respective communities. For standards communities, the artifact is a specification and participating organizations typically compete on implementations of that specification; however, the communities often have natural opportunities for cooperation through natural complementarities, such as between an equipment manufacturer and service provider. For open source communities, the artifact is a common implementation (in software), and so participants cooperate by contributing modules to the implementation but compete by selling complementary products and services beyond the common good created (West & Gallagher, 2004). General theories of participation in technical communities should be replicable across different organizing artifacts. In our case, we would expect that the trade-offs in sponsored open source communities would

also apply to sponsored standards communities, although the latter have only rarely been studied (see MacKie-Mason and Netz, 2004 for an exception).

A second major difference in the streams is the outcome measure. While the work of Rosenkopf and colleagues (2001) uses participation in technical communities to predict participant activities outside the community, most open source and other standards research focuses on processes within the communities. Some focus on the human capital of the communities (von Krogh, Spaeth, Lakhani 2003; Fleming and Waguespack, 2004), others on the governance and control processes within the communities (O'Mahony, 2003; Lee & Cole, 2003), while others look at the impact of external forces on the community activities (Simcoe, 2004).

Research examining internal community dynamics that uses community collaboration as antecedents can benefit by considering the actions of participants in terms of their shared and competing interests. The production of a public good depends on alignment of mutual interests, while the presence of direct (or indirect) rivals assures competing interests. Excluding direct rivals minimizes the risk of spillovers (MacKie-Mason and Netz, 2004) but limits the potential success of the community's activities in the broader marketplace.

Because firms collaborating with community-managed open source software projects contribute code that is also then available to their competitors, they have in a sense created a platform for pluralistic innovation. Firms who have chosen to initiate open source projects have chosen to "grow the pie" while claiming their slice of it: they are competing on a common platform. Prior research and industry practice have emphasized how firms accrued the benefits of platform dominance by establishing de facto standards for an industry (Morris and Ferguson, 1993; Bresnahan and Greenstein, 1999; West and Dedrick, 2000; West, 2003). However, with

the open source community based development model, no single firm either drives platform changes nor appropriates all their benefits — suggesting new directions for future research.

Future Research

We can see two ways in which future research on sponsored projects can improve our understanding of the relationships among firms, technical communities, and innovation. One approach is to compare the characteristics of a parallel population of sponsored and community led projects to contrast the antecedents, internal measures and consequences of sponsorship on such measures as project efficacy, sustainability and growth. We would expect to find the two forms to have some similar characteristics and causal relationships — but the role of sponsor control, resources and leadership needs to be examined. A second approach is to not only classify the motivations for sponsorship of open source projects (as West & Gallagher 2004 have begun to do), but, examine how firms sustain the competing goals of sponsoring a community project while simultaneously creating and capturing value for the firm. We would expect certain types of firm objectives to be more compatible with community goals than others, thus affecting the difficulty of achieving both goals simultaneously.

More broadly, we believe that research on technical communities in general should examine the degree to which the nature of the organizing artifact and the type of technology determines community structure and measures of efficacy. For example, Fleming and Waguespack (2004) describe a standards community (the IETF) that more closely resembles the open source community studied by von Krogh and colleagues (2003) than the standards community studied by Rosenkopf and colleagues (2001). Are there similarities in the founding events (or sponsorship structure) that account for these parallels — which would allow researchers to generalize across multiple types of communities? Or (as Bradner 1999 implies) is the IETF a

particular type of community and thus not representative of a broader class of community phenomenon?

This suggests an additional set of opportunities for research on technical communities. As a standards community, the IETF is unique in that individuals can directly participate without organizational affiliation or sponsorship, a model that later influenced the communities designed by open source software pioneers (Behlendorf, 1999). Despite the fact that the role lead users can play in the innovation process has been appreciated for some time (von Hippel, 1988), few technical communities studied by traditional technology management scholars have included individual participation.

Technical communities differ from the prior literature on consortia and standard settings bodies. Their variance in participation rules, scope of work, affiliation of participants, and formal and informal control mechanisms suggest opportunities to study relationships among these factors, and their impact on the development speed and adoption rate of the technologies they further. For example, certain structures (or cultures) might encourage more user innovation (von Hippel, 1987) and certain technologies may fare better with community development than others. At the firm level, compatibility of organizational cultures or even the personal needs of firm representatives might explain differences in firm participation and sponsorship of technical communities. Finally, more process research is needed to help explain the cultures and structures emerge to channel the divergent energies of both individuals and firms in technical communities.

Tables and Figures

Construct	State	Definition
Intellectual property	OSI-approved	Uses license approved by Open Source Initiative
	proprietary	Uses a proprietary license that does not meet OSI requirements
	dual license	Uses both OSI and proprietary licenses as a form of price discrimination
Development	open	anyone can participate without fee (although meritocratic processes may differentiate participation rights)
	gated	sponsors and those who pass sponsor controlled member process can participate
	closed	only sponsors' employees participate - employment relations guide project
Governance	community managed	individuals elected to leadership positions
	firm dominated	individual members have some rights but they are in the minority
	sponsor controlled	non-sponsors have no rights

Table 1: Dimensions of open source projects

Construct	State	Definition
Founding	organic	founded by individuals
	synthetic	sponsor founded (to meet sponsor goals)
Membership	I	individuals can join; may be employed by supporting firms, but vote as individuals
	F	firms joint as dues-paying members
	F+I	both firms and individuals can join as due paying members
	n/a	there is no membership structure

Table 2: Antecedents to community governance alternatives

Sponsor	Project	Type of Software	Start
MySQL AB	MySQL	SQL database	1995
Sun	Netbeans	Java IDE	1996
IBM	Jikes	Java compiler	1998
Netscape	Mozilla	web browser	1998
Apple	Darwin	OS kernel	1999
JBoss Inc.	JBoss	Java application server	1999
Novell	Evolution	personal information manager	1999
Jabber Inc.	Jabber	instant messaging	2000
Sun	OpenOffice	office productivity suite	2000
IBM	Eclipse	Java IDE	2001
OSAF	Chandler	personal information manager	2002

Table 3: Examples of Sponsored Projects

Area	**Role**
intellectual property	owns copyright in the resulting code
governance	sponsor control pre-empts community control
development	provide resources to assure ongoing development
founding	creates the project, exerts control, provides initial resources, possibly donate code
spinout	sponsor takes internal code and donates it to the project

Table 4: Roles of project sponsors

Dimension:	Comm	Dev	IP	IP	IP	Govern.	Govern.
Project	**Founding**	**Resources**	**Owns**	**Code**	**License**	**Members**	**Control**
Debian	organic	community	foundation	all	OSI	I	Community
GNOME	organic	community	foundation	all	OSI	I	Community
Apache	organic	community	foundation	all	OSI	I	Community
Mozilla	synthetic	spinout	foundation	all	OSI	I + F	Community
Eclipse	synthetic	spinout	foundation	all	OSI	I + F	Firm member dominated
Jikes	synthetic	spinout	sponsor	all	OSI	n/a	Sponsor controlled
sendmail	organic	spinout	sponsor	opening parts	dual	n/a	Sponsor controlled
Darwin	synthetic	fish bowl	sponsor	opening parts	OSI	n/a	Sponsor controlled
JBoss	synthetic	fish bowl	sponsor	partly open	dual	n/a	Sponsor controlled
MySQL	synthetic	fish bowl	sponsor	partly open	dual	n/a	Sponsor controlled
Chandler	synthetic	fish bowl	foundation	all	OSI	n/a	Sponsor controlled
Helix	synthetic	fish bowl	sponsor	opening parts	dual	n/a	Sponsor controlled
OpenOffice	synthetic	fish bowl	sponsor	opening parts	OSI	n/a	Sponsor controlled

Source: Interviews with project and sponsor representative
Key: See Table 1 for Founding, Control; Table 2 for Members; West (2003) for Code

Table 5: Classification of Sponsored, Community Managed and Hybrid Projects

References

Allen, R.C. 1983. Collective Invention. **Journal of Economic Behavior and Organization**, 4: 1-24.

Anderson, Philip & Tushman, Michael. 1990. Technological Discontinuities and Dominant Designs: A Cyclical Model of Technological Change. **Administrative Science Quarterly,** 35, 4: 604-633.

Apache. 2004. Management - The Apache Software Foundation. **Apache Software Foundation**, http://www.apache.org/foundation/roles.html.

Baldwin, Carliss Y. & Clark, Kim B. 2000. **Design Rules, Vol. 1: The Power of Modularity.** Cambridge, Mass.: MIT Press.

Baldwin, Carliss Y., & Clark, Kim B. 2003. The Architecture of Cooperation: Does Code Architecture Mitigate Free Riding in the Open Source Development Model? **Harvard Business School Working Paper Series, No. 03-209.**

Bijker, Wiebe E., Hughes, Thomas P. & Pinch, Trevor (eds). 1987. **The Social Construction of Technological Systems.** Cambridge, Mass.: MIT Press.

Bradner, Scott. 1999. The Internet Engineering Task Force. in Chris DiBona, Sam Ockman and Mark Stone, eds., **Open Sources: Voices from the Open Source Revolution.** Sebastopol, Calif.: O'Reilly, pp. 47-52.

Bresnahan, Timothy F., & Greenstein, Shane. 1999. Technological competition and the structure of the computer industry. **Journal of Industrial Economics** 47 (1), 1-40.

Chesbrough, Henry W. 2003. **Open Innovation**. Boston: Harvard Business School Press.

Christensen, Clayton M. & Rosenbloom, Richard S. 1995. Explaining the attacker's advantage: technological paradigms, organizational dynamics and the value network. **Research Policy** 24, 233-257.

Doz, Yves L., Olk, Paul M. & Smith Ring, Peter. 2000. Formation processes of R&D consortia: which path to take? Where does it lead? **Strategic Management Journal**, 21 (3), 239-266

Fleming, Lee & Waguespack, David M. 2004. Penguins, Camels, and Other Birds of a Feather: The Emergency of Leaders in Open Innovation Communities**. Harvard Business School Working Paper.**

Franke, Nikolaus & Shah, Sonali. 2003. How communities support innovative activities: An exploration of assistance and sharing among end-users. **Research Policy,** 32: 157-178.

Franke, N. & Von Hippel, E. 2003. Satisfying heterogeneous user needs via innovation toolkits: the case of Apache security software. **Research Policy,** 32.

Garud, Raghu., Jain, Sanjay., & Kumaraswamy, Arun. 2002. Institutional Entrepreneurship In The Sponsorship Of Common Technological Standards: The Case Of Sun Microsystems And Java**. Academy of Management Journal***,*** 45 (1), 196-214.

Hargadon, A.B. & Douglas, Y. 2001. When Innovations Meet Institutions: Edison and the Design of the Electric Light. **Administrative Science Quarterly,** 46: 476-501.

Hars, Alexander & Ou, Shaosong. 2002. Working for free? Motivations for participating in open-source projects. **International Journal of Electronic Comme***rce*, 6, 3, 25-39.

Healy, Kieran & Schussman, Alan. 2003. The Ecology of Open-Source Software Development. **Working paper, Department of Sociology, University of Arizona**, January 29, http://opensource.mit.edu/papers/healyschussman.pdf.

Hunt, Courtney Shelton, & Aldrich, H. E. 1998. The second ecology: Creation and evolution of organizational communities. **Research in Organizational Behavior** 20: 267-301.

"Interview with Dan Frey," *The Linux Line,* IBM, January 2003, URL: http://www-1.ibm.com/linux/linuxline/jan03/frye.shtml

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

Krishnamurthy, Sandeep. 2002. Cave or Community? An Empirical Examination of 100 Mature Open Source Projects. **First Monday,** 7 (6) URL: http://www.firstmonday.dk/issues/issue7_6/krishnamurthy/

Lakhani, Karim & Wolf, Robert G. 2003. Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects. September**, MIT Sloan Working Paper No. 4425-03,** URL: http://freesoftware.mit.edu/papers/lakhaniwolf.pdf

Lee, Gwendolyn K. & Cole, Robert E. 2003. From a Firm-Based to a Community-Based Model of Knowledge Creation: The Case of the Linux Kernel Development. **Organization Science,** 14, 6: 633-649.

Lerner, Josh & Tirole, Jean. 2002. Some Simple Economics of Open Source. **Journal of Industrial Economics***,* 52 (2), 197-234.

MacKie-Mason, Jeffrey K. & Netz, Janet S. 2004. Manipulating Interface Standards As An Anti-Competitive Strategy. **Proceedings of the Standards and Public Policy conference, Federal Reserve Board,** Chicago, May.

Millen, D.R., Fontaine M. & Mullerm M.J. 2002. Understanding the Benefits and Costs of Communities of Practice. **Communications of the ACM***,* 45, 69-73.

Moody, Glyn. 2001. **Rebel Code: Inside Linux and the Open Source Revolution***,* Cambridge, Mass.: Perseus.

Morris, Charles R., & Ferguson, Charles H. 1993. How Architecture Wins Technology Wars. **Harvard Business Review**, 71 (2), 86-96.

O'Mahony, Siobhán. 2002. The Emergence of a New Commercial Actor: Community Managed Software Projects. **Unpublished dissertation, Stanford University**.

O'Mahony, Siobhán. 2003. Guarding the Commons: How Community Managed Software Projects Protect Their Work. **Research Policy** 32, 7: 1179-1198.

O'Mahony, Siobhán. 2005. Non-Profit Foundations and Their Role in Community-Firm Software Collaboration. In Joe Feller, Brian Fitzgerald, Scott Hissam and Karim Lakhani, eds., **Making Sense of the Bazaar: Perspectives on Free and Open Source Software.** Cambridge, Mass: MIT Press.

O'Mahony, Siobhán, & Fabrizio Ferraro. 2004. Managing the Boundary of an 'Open' Project, **Harvard Business School Working Paper**.

OSI, 2004, Licensing. **Open Source Initiative**, URL: http://www.opensource.org/licenses/

Parker, Geoffrey & Van Alstyne, Marshall W. 2005. Innovation Through Optimal Licensing in **Free Markets and Free Software, Social Science Research Network**, January 3, URL: http://ssrn.com/abstract=639165

Pfeffer, Jeffrey & Salancik, Gerald R. 1978**. The External Control of Organizations: A Resource Dependence Perspective.,** New York, Harper & Row.

Pfeffer, Jeffrey. 1981. **Power in Organizations.** New York: HarperCollins.

Rosenkopf, Lori & Tushman,Michael L. 1998. The Co-evolution of Community Networks and Technology: Lessons from the flight simulation industry. **Industrial and Corporate Change,** 311-346.

Rosenkopf, Lori & Tushman,Michael L. 1994. The coevolution of technology and organization. in Joel Baum, and Jitendra Singh. (eds). **Evolutionary Dynamics of Organizations,** Oxford University Press: New York; 403 - 424.

Rosenkopf, Lori, Metiu, Anca & George, Varghese P. 2001. From the Bottom Up? Technical Committee Activity and Alliance Formation. **Administrative Science Quarterly,** 46 (4), 748-772.

Saxenian, AnnaLee. 1994. **Regional advantage: culture and competition in Silicon Valley and Route 128.** Cambridge, Mass.: Harvard University Press.

Schoonhoven, Claudia B. & Romanelli, Elaine (eds). 2001. **The entrepreneurship dynamic: Origins of entrepreneurship and the evolution of industries.** Stanford, CA: Stanford University Press.

Shah, Sonali. 2004.Understanding the Nature of Participation and Coordination in Open and Gated Source Software Development Communities. **Academy of Management Proceedings, New Orleans, LA, B1-6.**

Simcoe, Timothy S. 2004. Design by Committee? The Organization of Technical Standards Development. **Unpublished Ph.D. dissertation,** Haas School of Business, University of California, Berkeley.

Stallman, Richard. 1999. The GNU Operating System and the Free Software Movement, in Chris DiBona, Sam Ockman and Mark Stone, eds., **Open Sources: Voices from the Open Source Revolution.** Sebastopol, Calif.: O'Reilly, pp. 53-70.

Staudenmayer, N., M. Tripsas, & Tucci, C. 2000. Development Webs: A New Paradigm for Product Development. **In Winning Strategies in a Deconstructing World,** edited by M. Hitt, R. Bresser, D. Heuskel and R. Nixon. New York: John Wiley & Sons, Inc.

Stewart, Katherine J., Ammeter, Anthony P., & Likoebe M. Maruping. 2005. A Preliminary analysis of the Influences of Licensing and Organizational Sponsorship on Success in Open Source Projects. **Proceedings of the 38th Hawaii International Conference on System Sciences**, January 3 – 6, 2005.

Tripsas, Mary. 1997. Unraveling the Process of Creative Destruction: Complementary Assets and Incumbent Survival in the Typesetter Industry. **Strategic Management Journal***,* 18: 119-142.

Tushman, Michael L. & J. Peter Murmann. 1998. Dominant Designs, Technology Cycles, and Organizational Outcomes. **Research in Organizational Behavior***,* 20: 231-266.

Tushman, Michael & Rosenkopf, Lori. 1992. Organizational Determinants of Technological Change. **Research in Organizational Behavior** (14): 311-347.

Tushman, Michael L. & Anderson, Philip. 1986. Technological Discontinuities and Organizational Environments. **Administrative Science Quarterly** 31, 3, 439-465.

Välimäki, Mikko. 2003. Dual Licensing in Open Source Software Industry. **Systemes d'Information et Management.** URL: http://opensource.mit.edu/papers/valimaki.pdf.

Van de Ven, Andrew H. 1993. A community perspective on the emergence of innovations. **Journal of Engineering and Technology Management***,* 10, 23-51.

Van de Ven, Andrew H. & Garud, Raghu. 1994. The Co-Evolution of Technological and Institutional Innovations. in J. Singh and J. Baum (eds.) **Evolutionary Dynamics of Organizations,** New York: Oxford Univ. Press.

Van de Ven, Andrew H. & Hargrave,Timothy J. 2003. Converging Perspectives on Institutional Change in the Technology and Social Movement Literatures. **Working paper, Carlson School of Management,** University of Minnesota.

Van de Ven, Andrew, & Garud, Raghu. 1989. A Framework for Understanding the Emergence of New Industries. **Research on Technological Innovation, Management and Policy,** 4:195-225.

von Hippel, Eric. 1988. **The sources of innovation***,* New York: Oxford University Press.

von Hippel, Eric 2001. Open Source Shows the Way: Innovation By and For Users - No Manufacturer Required. **Sloan Management Review**, 42, 82-85.

von Hippel, Eric & von Krogh, Georg. 2003. Open Source Software and the 'Private-Collective' Innovation Model: Issues for Organization Science. **Organization Science,** 14: 209-223.

von Krogh, Georg, Spaeth, Sebastian & Lakhani, Karim R.. 2003. Community, joining, and specialization in open source software innovation: a case study. **Research Policy** 32, 7, 1217-1241.

West, Joel. 2003. How open is open enough? Melding proprietary and open source platform strategies. **Research Policy,** 32, 7, 1259-1285.

West, Joel & Dedrick, Jason. 2000. Innovation and Control in Standards Architectures: The Rise and Fall of Japan's PC-98. **Information Systems Research** 11 (2), 197-216.

West, Joel & Gallagher, Scott. 2004. Key Challenges of Open Innovation: Lessons from Open Source Software. **Working paper, College of Business, San José State University**, URL: http://www.cob.sjsu.edu/OpenSource/Research/WestGallagher2004.pdf.