The Role of Participation Architecture in Growing Sponsored Open Source Communities

Joel West
San Jose State University, College of Business
http://www.JoelWest.org/Research/OpenSource/

Siobhán O'Mahony
UC Davis, Graduate School of Management

Final manuscript draft
February 6, 2008

Published as
Joel West and Siobhán O'Mahony, "The Role of Participation Architecture in Growing Sponsored Open Source Communities," *Industry & Innovation,* 15, 2 (April 2008): 145-168.
http://dx.doi.org/10.1080/13662710801970142

Acknowledgements: Earlier versions of this paper were presented at the Academy of Management and European Academy of Management conferences. We thank the guest editors, two anonymous reviewers and especially the EURAM participants for their helpful suggestions.

Abstract: Most research on open source software communities has focused on those that are community founded. More recently, firms have founded their own open source communities. How do sponsored open source communities differ from their autonomous counterparts? With comparative examination of 12 open source projects initiated by corporate sponsors, we identify three design parameters that together help form a participation architecture – the opportunity structure extended to potential external contributors. In exploring sponsors' community design decisions, we found that sponsored open source projects were more likely to offer transparency than they were accessibility and that this had implications for their communities' growth. We contribute theoretical constructs that offer a common basis of comparison for the future study of open source projects and illustrate how the tension between control and growth affects open source community design and creation.

Technical communities often play a crucial role in helping firms to develop and deploy new technical innovations (Rosenkopf, Metiu and George, 2001; Rosenkopf and Tushman, 1998; Mowery and Simcoe, 2005; Fleming and Waguespack, 2007). Technical communities provide a vehicle for the exchange of technical information that fosters the accumulation of innovations (Saxenian, 1994; von Hippel, 1988; Allen, 1983) and enables actors from different organizational forms to collaborate (Rosenkopf and Tushman, 1998; Van de Ven, 1993). Technical communities, like industry associations, help firms identify mutual interests that might otherwise go undiscovered (Sabel, 1984). Empirical work suggests that firms benefit from participating in technical communities by gathering information on potential alliances, identifying opportunities for future inter-firm collaboration (Rosenkopf et al, 2001) and sharing risk (Tushman and Rosenkopf, 1992; Rosenkopf and Tushman, 1994; 1998).

One type of technical community that has received a great deal of empirical attention is an open source software development community. These communities are composed of individuals who collaborate toward a common goal but do not share a common employer and are not governed by an employment hierarchy. By using both online and offline means, open source software communities collectively produce software that is freely and publicly available – creating in effect a shared public good that can be used for either public or private purposes (von Hippel and von Krogh, 2003; Lee and Cole, 2003; O'Mahony, 2003, Murray and O'Mahony, 2007; Markus, 2007).

Prior research on open source communities has emphasized autonomous and "self-managed" communities that are typically founded by individuals or groups who recruit and mobilize other community members to contribute and grow organically. However, since the first corporate founded open source project (Mozilla) was launched in 1998, firms have begun to create or sponsor their own open source communities. Firms sponsoring open source communities typically due so as part of an intentional open innovation strategy (West and Gallagher, 2006). Because these communities are founded for strategic reasons, they are likely to differ from their autonomous counterparts. However, little is known about how corporate sponsorship affects how open source communities are designed and evolve. Too often the existence of technical communities is taken as a given, and the factors influencing their design unexplored (Hargrave and Van de Van, 2006).

If communities are an important vehicle for mediating firm interactions (Rosenkopf et al, 2001) and potentially for innovation outcomes (von Hippel, 2005; Jepperson and Frederiksen, 2006), then understanding such collaborations is crucial to any understanding of that roles that communities play in innovation. This research examines 12 sponsored open source communities, and contrasts them with prior research on autonomous communities. From this comparative analysis, we identified three design dimensions that corporate sponsors consider when designing open source communities: 1) intellectual property rights, 2) development approach, and 3) model of community governance. We found that design decisions in these three areas created a specific participation architecture: i.e. the socio-technical framework that extends participation opportunities to external parties and integrates their contributions. Much as architecture guides people in physical space, a participation architecture guides interactions and exchange in a community through the social, legal, and technical capabilities offered to community members. While prior research has shown that a project's technical architecture can affect community participation (Baldwin and Clark, 2006; MacCormack, Rusnak and Baldwin, 2006), there has been less appreciation for how community design choices can also affect participation.

By comparing the participation architectures that resulted from sponsors' design decisions, we identified two types of openness: transparency and accessibility. While transparency offered potential contributors the ability to follow and understand a community's production efforts, accessibility determined the degree to which external contributors could influence that production. In designing a community, sponsors were more likely to offer transparency than they were to offer accessibility to external community members. We found that sponsors faced a control vs. growth tension. To leverage the ability of communities to contribute to their firm's bottom line, sponsors sought to maintain control over the community's strategic direction. However, sponsors soon discovered that by restricting access to community processes, they limited their community's ability to attract new members and grow.

We contribute to the literature on open source communities, technical communities and firms and community collaboration in three ways. First, we identify some key distinctions between sponsored communities and autonomous communities that can help further research on firm-community collaboration and innovation. Second, we develop the construct of participation architecture and show how it is operationalized in a sample of open source communities. Third, we illustrate the "control-growth" tension that sponsors building communities face when making design decisions. Our research shows that participation in a community is determined not only by the technical architecture identified by Baldwin and Clark (2006), but also by the organizational structure that results from a sponsor's community building design decisions.

Firms and Technical Community Collaborations

To some extent, firms and technical communities have always collaborated (Allen, 1983) to create standards (e.g., Isaak, 2005), shared infrastructure (Bradner, 1999), and innovation outcomes (Hargrave and Van de Ven, 2006) that are bigger than any one firm can achieve. Empirical work suggests three reasons why firms participate in technical communities.

First, there is increasing evidence that path breaking innovations cannot occur without a community to interpret, support, extend and diffuse them (Hargrave and Van de Ven, 2006; 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). Research on technologies such as medical devices, bicycles, computer hardware, and electricity show that new technologies are shaped by human institutions that provide a context for the interpretation and use of such technologies (Bijker, Hughes and Pinch, 1987).

Second, research on collective models of innovation (von Hippel, 2005; von Hippel and von Krogh, 2003; Hargrave and Van de Ven, 2006; Allen, 1983) and on community technical organizations (Rosenkopf, Metiu and George, 2001; Rosenkopf and Tushman, 1998) shows that communities of lead users help firms not only in interpreting and applying new innovations, but in their creation and further development by refining new design iterations (von Hippel, 1988, 2005; Shah, 2006; Murray and O'Mahony, 2007). A large body of evidence suggests that firms in many industries (toys, entertainment, medical devices, manufacturing, sporting goods, music) benefit from contributions from community members (Jepperson and Frederiksen, 2006; von Hippel, 2005; Franke and Shah, 2003; Lakhani and von Hippel, 2003).

Third, although it is not widely recognized, technical communities provide a vehicle to coordinate the work of both firms and individuals in developing new technologies and standards (Rosenkopf et al, 2001; Mowery and Simcoe, 2005). Technical communities offer individuals leadership opportunities (Fleming and Waguespack, 2007; O'Mahony and Ferraro, 2007) and enhance their technical credibility (Hars and Ou, 2002; Lerner and Tirole, 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 (Fleming and Waguespack, 2007).

The degree to which firms benefit from collaboration with technical communities can also depend on features of the technology itself such as the degree of modularity (Baldwin and Clark, 2000), or the number of complements, linking mechanisms, or interface technologies required (Tushman and Rosenkopf, 1992). While practitioners have long argued that effective management of such communities can help support firm goals (Williams and Cothrel, 2000; Armstrong and Hagell, 1996; Godwin, 1994) little empirical work has been done in this area. To further our understanding of how firms and communities collaborate, scholars have recently turned their attention to a specific type of technical communities, open source software development communities.

Early research on open source software communities focused on individual motivations to participate. Lerner and Tirole (2002) argued that contributors to open source communities participate in order to improve the visibility of their skills in the open labor market. However, subsequent empirical work suggests that volunteer contributors to community projects are equally likely to receive intrinsic benefits. Contributors to open source projects do so because they enjoy solving technical problems, they identify with the project's goals, they are interested in building their skills, or simply want to improve the software for their own use (Lakhani and Wolf, 2003; Hertel and colleagues, 2003; Hars and Ou, 2002).

Since major corporations began incorporating community developed open source software in their products and services (e.g. West and Dedrick, 2001; Baldwin, O'Mahony and Quinn, 2003; MacCormack and Herrman, 1999), many communities have attracted donations of code and on-going development participation from firms as well as individuals (O'Mahony, 2002; 2005; Dahlander and Magnusson, 2005). Because a community's output is publicly available, it limited the extent to which a firm's direct benefit from their investment in the community would remain unique to them. However, firms gained indirect economic benefits and competitive advantage by leveraging the widespread adoption of popular open source projects, leveraging the shared R&D investment of the community, and selling other goods and services necessary to provide a complete solution (West, 2003, 2007; West and Gallagher, 2006).

Some communities explicitly recognize firms as participants, while in other cases firms participate indirectly through employees (i.e., sponsored contributors) that represent the firm's interests (Dahlander and Wallin, 2006). Often, an open source community will create an informal or formal social structure to manage membership and joining (von Krogh, Spaeth and Lakhani, 2003; O'Mahony and Ferraro 2007), but little has been done to understand how these projects are governed (see Shah, 2006; Markus, 2007; O'Mahony, 2007 for recent exploratory exceptions).

In addition to collaborating with autonomous open source communities, a growing number of corporate and government sponsors have founded their own open source communities to meet either public or private objectives. West and O'Mahony (2005) distinguished between individually-founded and organizationally-founded open source communities, designating the former as "organic" and the latter as "synthetic". These distinctions emphasized the different character of these two types of community founders and their prospective growth strategies. While organic projects are founded by individuals and grow through grass roots communications, synthetic communities are founded by corporations and grow with more strategic direction. Since communities can evolve along different trajectories after their founding, in this paper, we refer to *autonomous* and *sponsored* communities to focus on their current governance structure (cf. Markus, 2007; O'Mahony, 2007) as opposed to their founding state. For example, a synthetic community could begin as a sponsored community, but evolve to become fully autonomous – as was the case with Netscape's release of Mozilla and its subsequent transition to independence. By using the categories of autonomous and sponsored, our aim is to provide a more precise categorization to help scholars examine open source communities over their lifecycle.

We define an *autonomous open source community* as one that is presently independent of any one firm and community managed (cf. O'Mahony, 2007). A community-managed governance system operates outside the reach of authority embedded in employment relations. Contributors to an open source project may be volunteers or may be paid by their employers to work on the project, but decision-making on the project takes place independently from the employment structure that guides the workplace. These projects may be supported by a non-profit foundation created specifically to support the project, but such foundations have little authority over their members (O'Mahony, 2005).

A *sponsored open source community* is one where one (or more) corporate entities control the community's short- or long-term activities. To refine these distinctions, in thus study, we examine how sponsors approached the task of building an open source community and how these communities differed from their autonomous counterparts.

Methods

Given that there has been relatively little research on sponsored open source software communities, we examined why sponsors created open source communities and how their motivations affected the design of the communities they founded. Because sponsors understood that the design of an open source community affected a community's ability to attract contributors, and because this was widely regarded by sponsors as a sign of a successful open source community, we asked: how do community design decisions affect their ability to attract external participants? Since our informants frequently referred to autonomous communities as the inspiration for their own community building efforts, we analyzed the data in comparison with findings from prior research on a limited number of previously-studied autonomous communities For comparison purposes, we contrasted our findings with the earliest and most successful autonomous communities that had the greatest influence on the decisions of our firm sponsors: Apache and four major Linux-related communities.[1]

Research Design. We adopted a grounded theory approach, which is well suited for phenomena that are emergent or poorly understood (Strauss and Corbin, 1990). Such an approach provides the broadest possible contextual information for understanding a phenomenon where there is not strong *a priori* theory (Edmondson and McManus, 2007). Thus, the study was guided by an inductive, qualitative approach using ethnographic methods.

Sample. To study sponsored open source communities, we employed a theoretical sampling approach to identify a wide range of possible structures and relationships (cf. Glaser and Strauss, 1967). We selected 12 sponsored open source communities founded between 1983 and 2004 that embraced elements of the open source model: an open source license, publicly available code and evidence of community on mailing lists from. Since there was no single source available that could identify the population of sponsored open source projects, we drew upon our prior field work in the open source field to identify such a population.

We excluded from our sample of study, those projects not perceived as open source by open source developers, such as Microsoft's "shared source" projects which provide a subset of open source intellectual property rights to a defined sub-groups of customers (West and Dedrick, 2001) or "gated communities" that use open source development processes for a defined population without public release of intellectual property (Shah, 2006). Of the sponsored open source communities that we identified and met this criteria, six were founded by traditional large proprietary technology companies, five were founded by firms to commercialize the open source software produced by the community, and one was sponsored by a non-profit organization set up specifically for that purpose. Table 1 presents descriptive information on the 12 projects selected.

Data. As this was exploratory research, we interviewed one to six informants from each sponsoring organization. After selecting communities, informants were identified from project webpages and industry conferences. Semi-structured interviews were 60-90 minutes long, and focused on understanding how sponsors approached the prospect of building community. More specifically, interviews covered the following domains: 1) when and

[1] Because our focus was on sponsored communities, we did not gather primary data on autonomous communities, and thus our reference group was limited both in size and diversity. We believe that group was representative of early, successful autonomous projects but not of a broader population of autonomous projects that include both successes and failures.

how sponsor founded an open source community; 2) how they prepared for and designed an open source community; and 3) their experience thus far in building community. Most of the interviews (N=23) were conducted face to face, with the remainder (N=6) over the phone. These visits were followed by an analysis of the community's website and follow up questions posed to the informants as appropriate. We collected our data from early 2002 through mid-2005.

Data Analysis. We analyzed the interview transcripts, coding relevant observations compared and contrasted our interview notes. Emergent themes in the data were synthesized in research memos written in an on-going research log. As our research progressed, we contrasted our interim findings with prior research on autonomous communities to better understand the distinctions between the two.

Findings

While prior research on open innovation has found that open approaches to developing communities can vary in the degree to which they are "open", our informants taught us that when forming a community, there were two distinct types of openness: *transparency* and *accessibility*. Transparency allows outsiders (i.e. non-sponsors) to understand what is happening and why — and, in the case of an open source community, allows use of the community's final product, the source code. Accessibility allows external participants to directly influence the direction of the community to meet their specific wants and needs, regardless of whether the external party is a hobbyist or an organizational adopter. In some cases, external contributors could be sellers of goods and services that might either compete with or complement the sponsor's business.

How were these two different types of openness provided? By comparing our data across 12 sponsored open source projects, we found that the design choices made by sponsors of open source communities could be categorized into three dimensions: 1) the organization of production; 2) community governance and 3) intellectual property. We identified these conceptual categories by examining how sponsors' design choices affected what community participation. Decisions with regard to the organization of production shaped how code development would took place, while community governance decisions shaped the processes by which decisions were made within the community; and intellectual property design decisions affected the allocation of rights to use the community's output. Table 2 shows how these three design dimensions compare with those of proprietary software development, the approach historically used by companies in the software industry.

After noting the importance that our informants placed on designing community forums for participation, we realized that, when considered together, these three dimensions of community design formed the basis of a participation architecture. We define a participation architecture as the socio-technical framework that extends opportunities to external participants and integrates their contributions.[2] A participation architecture guides interactions and exchange in an online community and encompasses the social, legal, and technical capabilities offered to community members. While sponsored communities were by some measures less open than most autonomous communities, in all cases, the decision to create an open source community was inherently a more open approach than a proprietary software development approach.

Within these three design dimensions that we categorized, we identified 11 design parameters chosen by the founders and subsequent leaders of the communities in our sample (Table 3). Both transparency and accessibility were relevant across each of the three dimensions of an open source project. After coding our data to understand whether a design parameter affected the organization of production, governance or intellectual property,

[2] The concept of an "architecture of participation" was initially articulated by O'Reilly (2005) as a set of heuristics that encourage participation and innovation. The "participation architecture" construct we develop here is intended to meld that earlier usage with an interorganizational analog to the "technical architecture" of Baldwin and Clark (2006).

we revisited it to determine whether that design choice enabled transparency or access.

We found that the degree to which sponsors' offered transparency or accessibility in these key areas affected sponsors' ability to attract external participants and grow these communities. By examining sponsors' community design decisions, we discovered that one of the primary challenges sponsors faced was how to how manage the tension between controlling the community in order to leverage their investment in it and opening up access to the community in order to attract greater growth of participants.

Managing the Tension Between Control and Openness

One key difference between autonomous (community managed) and sponsored open source software projects is that the sponsors of open source projects faced a fundamental tension between two conflicting goals. On the one hand, sponsoring an open source project was intended to advance the goals of the sponsoring organization. Sponsoring an open source project required significant investment in preparing the code, hosting the site, providing introductory materials and marketing the new opportunity. As such, community sponsors sought to maintain some degree of control over the project to assure ongoing alignment between their investment in the community and related product goals.

On the other hand, the provision of source code under an open source license was an inherently open approach intended to win greater external participation and technological adoption. In some cases, sponsors sought adoption from prospective users (cf. West, 2003); in other cases, they sought adoption from producers of complementary products or even direct competitors.[3]

We found that for the most open communities, the participation of external parties provided sponsors with both direct benefits (such as code contributions and bug reports from participants) and indirect benefits (such as marketing and adoption benefits from their open approach). For the most closed communities, sponsors thought that the primary benefit they received from creating an open source community was not from direct community contributions, but instead from increased public awareness, accelerated low cost distribution, and reduced costs of marketing.

The CEO of one open source startup explained how expensive commercial marketing channels were relative to the marketing benefits that could be derived from an open source community.

> Every dollar you give [proprietary competitor], 70 cents to goes to fund the sales and marketing efforts, and maybe 11-14% actually goes to pay the engineering salaries that write the code. …The barriers to entry into this marketspace isn't building a better product, it is having $50-60 million a year just to blow on sales and marketing. And that's really a shame. I just thought that was really frustrating from someone that is an innovator, someone that wants to compete by

3 Comparing the size of the community (or amount of community participation) between our 12 communities is problematic because the product category for some projects (e.g. Mozilla) have much bigger potential audiences than others (e.g. Sendmail). Such comparisons are made more difficult by the relative scarcity of competing open source projects within the same category. Thus we rely on each project's sense of its growth relative to the inherent potential for such growth, including the statements by some informants that they made choices to restrict openness that they knew limited the growth of contributions or other forms of participation.

> building a better product [rather than] on just the sheer economics of sales and marketing.

While sponsors shared a surprising degree of agreement on their motivation to create an open source community and the most immediate resulting benefits, they made very different design choices with regards to the type and degree of openness with which they were most comfortable. Our informants taught us that there were two kinds of openness to consider when designing an open source community: transparency and accessibility. Both transparency and accessibility were relevant across each of the three dimensions of an open source project (Table 2).

Transparency. Transparency meant that the code was publicly available, that most of the software production process was discussed on public mailing lists or discussion boards, and that the software release cycle and goals were also provided on the community website. While this was fairly unproblematic for static information about a project, the transition to a transparent production *process* took some getting used to for most sponsors. After receiving complaints that they were not being transparent enough in their development process, one community sponsor reevaluated what needed to be public and what needed to be private, deciding to err on the side of transparency. In making that decision, they decided to limit only private discussions only to those involving third parties — but this required changing ingrained habits. As its community manager explained:

> It [becoming open] took a while. We talked about it before. Then we got a complaint from someone working with us, one of the contributors saying I think I'm missing stuff. I don't know exactly how justified that particular complaint was. I am not sure that I agreed with that.….Then we looked at the development threads, and said, "Well what of this really needs to be private?" "The third party stuff", and then we said, "Okay let's do it [be open]." So some of it I think was timing, as people get more used to being open. We went through a very similar thing at [another sponsored project], killing the internal lists….So we renamed them with long, ugly, awkward names. Of course you might get auto-fill or something, but if you just naturally type, you got the public list. If you wanted to do something private it was hard instead of easy. But because a lot of that wasn't malfeasance, it was just the habit of doing what you do.

This sponsor renamed internal mailing lists to encourage developers to default to public lists and change the base of operations from private to public: a technological change to instigate a cultural and process change. While the sponsor reported that there were one or two human resources issues that ended up on the list that probably should not have, the changes were largely viewed as positive for both the organization and the community, and resulted in the development of a new reporting tool that could simultaneously be used for both public and private purposes.

Accessibility. The second type of openness sponsors considered when designing a sponsored community was the degree of accessibility — the amount of control sponsors would relinquish to the community. An accessible community not only provides visibility, but allows some outsiders to gain access to either the project's code repository, community planning processes, or strategic decision-making. Much like a community founded by individuals, the decision to provide access at different levels was usually phased and based on community members' demonstrated competence and evidence of contribution. However, sponsors supporting even the most accessible communities were likely to retain control over rights allocation rather than devolving it to the community. Two sponsors delegated control of code commit rights for subprojects to external parties, but retained control of higher level decision-making.

In this way the organization of production and governance became linked (to be discussed further in the next section). In doing so, sponsors explicitly recognized that granting accessibility triggered the loss of some corporate control in a way that a purely transparent model did not — particularly when external participants were from other (possibly competing) companies. In a community that did not yet have a formally approved set procedures for decision-making right allocation, these issues were constantly negotiated among contributing firms. As another community manager explained:

> If you are building an Open Source project, which has several commercial players at a significant level, you do a lot of negotiating and a lot of figuring out, "Well what is the right milestone schedule?…What is the direction of the code base? And what time frame? Who can contribute code? How do you decide what code is good enough? Who controls, or how do you control the relationship of the [project] release to those of commercial products?

In general, sponsors who most valued outside code contributions were more likely to offer accessibility to outside community members, recognizing that only by devolving some level of control, could they hope to attract the most talented programmers outside the firm.

All sponsors worked to achieve significant transparency in their open source communities, but sponsors varied considerably in the importance they placed on providing accessibility to external parties. This distinction provides a more nuanced understanding of the tension between openness and control. To make this abstract tradeoff more concrete, we identified three specific community design parameters that affected the degree of transparency and accessibility that sponsors provided.

Organization of Production

When forming an open source community, sponsors borrowed heavily from the tools and techniques pioneered by autonomous open source communities. This included general-purpose online tools such as web pages and particularly e-mail discussion lists. However, most also used some of the tools specifically developed and refined for open source software production, notably the CVS source code control system and the Bugzilla error tracking database (Robbins, 2005).

On the technical side, one key enabling element is modularity. Baldwin and Clark (2005) show how a modular architecture offering design options increases a developer's incentives to join an open source community and remain involved. Architectures that are modular allow developers to focus their talents on specific modules without having to learn the whole system (Baldwin and Clark, 2005). By maintaining compatibility with design rules within modules developers can self-select the modules they know best, reducing participant learning curves and thus lowering the cost to participate (Baldwin and Clark, 2000).

Sponsors were aware of how the degree of modularity could affect potential barriers and costs to participation. Thus, sponsors invested significant resources in creating or increasing modularity, consistent with what MacCormack, Rusnak and Baldwin would predict (2006). For example, the founders of one project rewrote its version 3.0 to create more modular interfaces than previous versions — to make code development by others easier and facilitate modular extension by community members. Another dual licensed project's highly interdependent architecture provided high performance but made it nearly impossible for outside participants to become proficient with the code — thus inhibiting potential community contributions. In contrast, the sponsor of one of the newer communities emphasized that "we wrote the code knowing it would be read"; they provided code reviews and other quality improvements to explicitly attract outside contributions within months of its first release. Another sponsor bragged that his was the "best documented open source project."

Overall, the degree of modularity, associated dependencies, and the quality of code documentation affected the ability of outside members to understand the code well enough to contribute. However, many of the measures available for individual projects were not comparable. In addition to the technical architecture of the code, the organization of production includes control of the processes by which individuals participate in the community's production process. These social measures are not necessarily correlated to a project's technical design: for example, highly modular code can still be tightly controlled by a single firm. Thus, a project's technical architecture is one subset of a community's participation architecture. As the rest of the paper will show, to more fully understand how both social and technical attributes affect the opportunities for others to participate requires consideration of all three design parameters – the organization of production, governance and intellectual property. We identified three design parameters that provided contributors with transparency and accessibility to production processes:

1. *Live code access* provides transparency by offering the community the chance to review the most recent "live" version of source code on the community website — which is more likely to have bugs than a finished product. Nearly all of the sponsored communities (10/12) allowed external participants to anonymously access the most current source code, subject to the sponsor's license terms. External contributors were thus able to follow the community's development cycle and contribute bug reports if they were so inclined.

2. *Public commit process* refers to the opportunity for community members to become directly involved in the production process by earning (through demonstrated technical proficiency) the right to directly commit software changes to the community repository. While all of the autonomous communities provide such accessibility, only some (5/12) of the sponsored communities did so. Seven of the sponsored communities did not publicly explain how one could go about acquiring commit rights to their projects, but most of them (6/7) encouraged people to send code patches via email.

The lack of information about gaining committer rights is not necessarily an inhibitor for participation but it limits the status and influence that a contributor can achieve. As one community member explained, "people can be phenomenally valuable contributors without having access for a long time. Somebody else can check their code, but people don't like it. It is seen as a mark of belonging." Sponsors that did not provide commit rights recognized such rights would increase participation but were unwilling to relinquish that much control over their code.

3. *Subproject creation* is a mechanism by which a community based on the sponsor's original code can grow to assume new functionality or new directions. We defined subprojects as new "start-up" projects that were allowed to govern themselves and address unmet needs independent of the larger community; allowing creation of such subprojects provides accessibility by decentralizing control over growth and innovation in the community. While all of the autonomous projects allowed community members to autonomously propose subprojects based on their own initiative, only 5 of the 17 sponsored communities did so.

Assessing Openness. Source code access offers potential external community participants transparency into the development process which can help them learn how the code is developed. Without awareness of the community's production process, the learning curve to make a meaningful contribution and thus gain membership or access within the community will be limited. The ability to create subprojects offers access to external parties by providing them opportunities to shape the future direction of the project. This type of access sends a powerful signal as to the ease with which external contributors can get new ideas proposed and accepted. Allowing individuals the right to earn commit rights offers the ultimate degree of accessibility – the ability to make direct contributions to the code. As Table 3 shows, more sponsors were more likely to provide transparency in their production processes than they were to provide accessibility.

Governance

We define openness in open source governance as the amount of decision-making control that sponsors relinquished to the community. For many in our sample, divesting some degree of control was as much a legitimating strategy as it was a recruitment strategy. To attract talented contributors, sponsors thought they both needed to acquire legitimacy in the open source community at large, and provide skilled participants the opportunity to take on greater responsibility in leadership roles. As one community founder from a Fortune 100 firm explained:

> [P]art of the message behind open source is that it is open and the community makes the decisions. And if what we actually did was said that the community makes decisions, but in practice, [the sponsor] makes all of the decisions then they would say "well this is not real".

To provide both leadership and legitimacy, some communities have a formal concept of membership, vesting members with key governance decisions, and are thus more accessible to participants. For other communities, *de facto* control remains with founding individuals (due to superior legitimacy or technical knowledge) or with founding firms (that provide the bulk of ongoing resources). The design levers available for managing openness in governance include:

1. Nonprofit foundations. All five individually founded communities and two sponsor-founded communities created a formal, legal, non-profit foundation to help manage community governance and assets; however, for historical reasons, the Linux foundation plays a more limited role than the others. As O'Mahony (2003, 2005) identifies, such foundations provide institutional permanence independent of any one individual, as well as legal status to negotiate with external entities (largely for the provision of resources). Because the creation of a non-profit foundation requires a board of directors and regular meetings, the introduction of such organizations also increased the transparency of control of the community assets.

2. Membership. Four out of five of the individually founded open source communities have formal processes by which an individual is recognized as a member; one community recognized both individuals and firms as separate classes of members. Only one sponsor founded community project created a nonprofit foundation with a membership base, while two others were under development at the time of our study. After offering external contributors membership rights, this sponsor experienced unprecedented growth in participation. Communities with a membership base provide members with some voice in formal governance matters (typically through annual elections and the right to vote on project wide decisions such as license or name changes), while non-membership communities remain either fully sponsor-controlled or retain *ad hoc* governances mechanisms.

3. Membership fee. Two communities (one individually founded and one sponsor founded) obtain funds from interested firms by selling memberships at a range of prices. In general, membership fees were not adopted by sponsors trying to create communities, but by those sponsors trying to create commercial ecosystems.

4. Release authority. The ultimate test of an online production community is who makes the final production decisions. For an open source community, this decision occurs when software is released. This authority may vest in the members, the affiliated foundation or remain with the sponsor or individual founder. Again there is a dramatic differences in openness between autonomous and sponsored projects: the community holds authority in all but one autonomous community (Linux), but only in two of the 12 sponsored communities. Such limited accessibility appears to reflect the sponsor's desire to align the features, quality and schedule of open source releases to its commercial goals.

Assessing Openness. By creating a membership organization and the opportunity to elect leaders, a few sponsors offered potential contributors the ability to develop a sense of belonging and become more vested in the community's future. The ability to gain "membership status" was viewed as a motivational and recruitment tool. Sponsors who adopted these community design features did so with the belief that divesting some control was necessary in order to attract talented contributors. However, most sponsors did not create an independent form of governance, retained exclusive release authority, and final say on all key community decisions.

Intellectual Property

Sponsored open source communities faced a potential barrier to external participation not found in their autonomous counterparts: the fear that the founder's desire to profit from community based production would limit the benefits that accrued to contributing members. West (2003) refers to this as a producer's inherent tradeoff between winning adoption of a technology and appropriating the returns from that technology. For open source communities, the key attributes of this tradeoff were associated with the ownership and licensing of the community produced software:

1. Content ownership. Most communities that created an affiliated foundation also vest ownership in the foundation (Linux an exception), as did two sponsored communities. For the other sponsored communities, ownership remains with the sponsor. Ownership of the content by a foundation provides a credible assurance to community participants that the code will remain available to participants in perpetuity (O'Mahony 2005). The direct ownership of the code by sponsors in most of the communities was a clear direct and symbolic measure of their design to maintain ongoing control over the terms by which outside participants access the code.

2. Subproject ownership. For the projects that allowed creation of new subprojects, the conditions of access were not necessarily the same as for the core code. For the five sponsored projects that allowed subproject creation, in four cases, ownership of the subproject output was the same as for the core project (two owned by the sponsor, two owned by the foundation); for one sponsored project, the output was owned by the contributing community members.

3. Software license. 10 of the 12 sponsored communities use one of the more than 50 licenses approved by the Open Source Initiative. As Lerner and Tirole (2005) and others have noted, the most popular open source license is the GNU General Public License (GPL). This widespread use and popularity among potential participants influenced the decisions of two sponsors — Mozilla and Helix — to offer their product under both their own license and the GPL, while other sponsors mentioned the importance of using a license deemed compatible with the GPL or the Lesser GPL (LGPL). Two sponsors used licenses that are as yet unapproved by the OSI. One community (Sendmail) was founded using an approved license (BSD) but the sponsor subsequently chose to release a major update under a non-approved license that discriminates against for-profit users. The last (Sugar) created its own license (adapted from the Mozilla Public License) that required publicity for their technology when used by service providers.[4]

4. License type. Open source licenses such as the BSD or Apache license allow recipients to use the code largely without restriction, while "free software" licenses (notably the GPL) compel recipients to return any modifications or changes (West, 2003). Rosen (2004) classifies these two types as "permissive" and "reciprocal"; a few licenses (such as the LGPL) impose reciprocal obligations on part but not all of the code. Four of the sponsored communities in our sample use a dual license strategy to implement price discrimination, with non-profit users selecting the free reciprocal license (i.e. the GPL) and for-profit users customarily paying to use the software without the reciprocal obligations (cf. Välimäki, 2003).

Assessing Openness. The ownership of code was the most dramatic difference between autonomous and sponsored projects, in that the sponsor in nearly all cases retained ownership of the core (if not subproject) code. A subset of the sponsors used the objectionable restrictions of the reciprocal license to provide revenues through a dual license policy, but otherwise the sponsored and autonomous communities

[4] Sponsors of the two unapproved open source licenses (Sugar, Sendmail) argued that their licenses largely met the requirements of the Open Source Initiative, although they did not submit them for OSI approval. Their use of unapproved licenses they termed "open source" stimulated occasional controversy within the open source social movement, but did not seem to impair the effectiveness of their community strategies. In December 2007, SugarCRM solved this problem by releasing its free software under GPL Version 3, an OSI-approved license.

followed similar license policies, either using standard or their own open source licenses.

Discussion

We defined and contrasted two different types of open source software communities: those sponsored by corporate organizations and the more traditionally studied autonomous (community managed) communities. Our study had two research questions: how did sponsors design open source software communities in the hopes of attracting external participation, and how did this differ from the design of autonomous based communities?

By studying the design decisions that sponsors made when creating a community, we identified three dimensions that affected participation: 1) the organization of production, 2) governance, and 3) intellectual property. In doing so, we showed that the participation architecture of a technical community is determined not only by its technical architecture, but also by community design decisions made by the community's leaders. While modularity in the technical architecture remains important to enabling participation by reducing the learning curve or cost of entry (e.g. Baldwin and Clark, 2006), the aspects of community design that we identified are also critical to attracting and enabling participants primarily because they shape the landscape of opportunities extended.

We showed that sponsors' community design decisions on these three dimensions reflected the inherent tension between two conflicting goals. On the one hand, firms wished to retain control over technologies fundamental to their business success. On the other hand, providing the opportunity structure for others to participate was a prerequisite for gaining the benefits from developing an external community. Thus, when designing a participation architecture, firms mediate between surrendering control and offering opportunities for outside participation that could lead to community contributions and growth.

The Role of Participation Architecture in Growing Sponsored Communities

We discovered that before designing their own open source software communities, our informants studied well known autonomous communities in some detail and made frequent reference to them. Thus, in the presentation of our findings, we compared the findings from our sample to some of these well-known autonomous communities.

Based both on qualitative data from our informants and online data from the communities in our sample, we found a fundamental tension unique to sponsored communities: while sponsors recognized that the key to attracting and retaining participants to their communities was to provide unfettered opportunities for contribution, they had an interest in retaining some controlling influence over the communities they founded to ensure these communities remained aligned with corporate strategy. Managing this tension was a pervasive concern and illuminates some of the challenge in using external communities to pursue open innovation. After identifying the tension between openness and control, we identified more precisely how sponsors reconciled this tension with 11 specific community building design parameters that cluster across three dimensions.

From our data, we found that the sponsor-founded communities could be classified into one of three distinct groups as sorted in Table 3. The first group of firm-created communities either had achieved or were seeking levels of community participation comparable to those of individually-founded communities; in fact, two of the communities (Eclipse, Mozilla) transitioned from corporate sponsored to become independent autonomous communities.

At the other extreme, a second group of communities (the three firm-sponsored dual-licensed communities) offered what we term a "fishbowl" development pattern — with the sponsor offering transparency to outsiders, but not accessibility to software development. A third group of communities lay somewhere in between: experimenting with the provision of access but not willing to give up key points of control.

Strikingly, sponsors were far more likely to provide transparency than they were accessibility, despite the possibility that a more controlled governance structure offered fewer opportunities for leadership and could thus reduce the sponsor's ability to recruit contributors. Community design decisions to provide either transparency and accessibility had very different effects. Transparency was cited by informants as critical to aiding adoption of the software: a key goal of all sponsors. The effects of accessibility were more mixed: while accessibility could potentially enhance the volume and quality of contributors to a project, it could also compromise sponsors' control over production. For when development was made fully accessible to external parties, more parties to decision-making created new dependencies and coordination costs for software that was critical to firm product lines.

However, there was no direct evidence of the direction of causality between the provision of accessibility and external community participation. Firms offering less accessibility could be motivated by a need to retain control, a belief that there was no benefit to doing so (because the community would not help in production anyway), or perhaps diminished expectations of external participation that became a self-fulfilling prophecy. Accessibility is only one of the factors that drive participation, as the high rate of participation (and adoption) for the tightly-controlled MySQL community would suggest.

Most of the sponsored communities produced software of interest to a large potential audience of user-adopters. Except for the dual-license software communities, there seemed to be little relationship between a sponsor's license choices and the overall accessibility of the community. However, communities that were less accessible (Darwin, MySQL, Sendmail, Berkeley DB) seemed to be due to a sponsor's a stronger need for control due to a greater fear of cannibalization of core revenues. Conversely, four of the most accessible sponsored communities (Eclipse, Mozilla, OpenOffice, Helix) produce software facing intense competition against a well-funded proprietary alternative, and thus these sponsors were most concerned with attracting external collaborators to aid in production and adoption. The participation architecture offered by these communities most closely resembled autonomous communities.

While a few sponsored communities sought outside participation by emulating key accessibility characteristics of autonomous projects (Apache was cited as a notable model), informants suggested that both creating accessibility and attracting significant external resources was a long and difficult process. Sponsors often approached the challenge in phases, offering transparency while preparing for accessibility. As one community manager trying to guide his community from a transparent model to a more accessible one explained, "the transparency and the communication of what is happening is a prerequisite for almost anything else".

<u>Contrasting Sponsored and Autonomous Communities</u>

Autonomous open source software communities have received a great deal of empirical and scholarly attention within the last decade. However, there has been very little research on corporate sponsored open source communities, on how they differ from autonomous ones, or on how such communities contribute to a firm's open innovation strategy. This research takes a first step towards answering these questions. By comparing sponsored communities and autonomous communities we found some important commonalities. Both offer access to code that is guaranteed by an open source license, which fits the definition set by the Open Source Initiative. Both also offer a high degree of transparency of access to that code — without which the rights to use the code would be useless.

However, our study showed that sponsored open source software communities are fundamentally different from autonomous communities in the potential for goal conflict between sponsor and community members. Although both sponsors and members seek widespread adoption, the primary goal of a corporate sponsor is profiting from its investment, while the goal of an open source community would be improving the capabilities of the shared technology.

To gain interest from a community of contributors, sponsors needed to at least provide transparency. The openness of sponsored communities differed most in terms of accessibility, with most sponsors retaining a privileged (monolithic) rights for some portion of the community's decisions. In a few open cases, the sponsor shared some control with the community — and when sponsors relinquished more control to the community, those sponsored communities were transformed into autonomous ones.

As consequence, we also found a dramatic difference between most sponsored and autonomous communities in terms of design decision related to accessibility, particularly in terms of governance. Governance of autonomous projects was largely pluralistic, shared widely among community members, whereas the ultimate decisions of sponsored communities were (with rare exceptions) controlled by the sponsor.[5]

The dichotomy is not complete, because not all autonomous open source projects provide full accessibility. Raymond's (1999) stylized typology of "The Cathedral and the Bazaar" contrasts the tightly-controlled BSD projects with the more open Linux. However, today the "cathedral" archetype is relatively rare: Raymond's criticism (and the success of Linux) have meant that successful autonomous projects have largely followed Linux in granting accessibility to potential contributors. However, as a practical matter the importance of community contributions constrains the accessibility decisions of autonomous communities more than sponsored ones: independent communities that don't attract contributions will have trouble producing new software, while sponsors can (and do) sustain communities with their own resources — as happened with MySQL and Berkeley DB in our sample.

[5] We use as our "control" group those large, successful pluralistic autonomous projects best known to our informants. Like our informants, we are thus drawing inferences from best practices rather than a cross-section of autonomous projects, which limits our ability to draw contrasts. For example, those autonomous projects that are less successful in attracting an external community would have less developed formal governance than the successful projects listed in our control group.

Future Research

Our comparative analysis of sponsored open source software communities and identification of the theoretical constructs that affect community design should enable future comparative work in community innovation. Our inductively generated framework can help scholars explicate and articulate differences and similarities across technical communities involved in the wide range of production of shared information goods, whether such goods are software, reference data (e.g. Wikipedia) or a travel guide (World66). Future research would do well to quantify how the design of a participation architecture affects a peer production community's growth trajectory.

Our study focused on corporate sponsored open source communities. However, we recognize that an increasing number of organizations such as private non-profit foundations, governments, and even transnational organizations sponsor technical communities. We would expect that communities sponsored by government or nonprofit actors would be more likely to favor public good ahead of the sponsor's pecuniary gain, but face similar tensions between maintaining control and attracting community participation and growth.

Firms have long sponsored external communities of users (such as chat rooms or bulletin boards) to provide communication with users, to both diffuse new technology and obtain user feedback. Sponsorship of external communities has been used as a source of open innovation, whether in musical instruments (Jeppesen and Frederiksen (2006), computer games (West and Gallagher, 2006; Prügl and Schreier, 2006) or sporting goods (Franke and Shah, 2003). Open source software has been held up as an exemplar of the process of user-contributed or open innovation (von Hippel, 2001, 2005; West and Gallagher, 2006). Certainly, the proliferation of autonomous open source projects has brought a raft of experimentation and proliferation of community forms. However, the growing popularity of sponsored communities suggests that firms can also sponsor open source communities and attract external participants. The question that remains open is under what conditions are firms more likely to sponsor such communities? Are these communities alternatives or complementary to in-house software development activities? How do they relate to a sponsor's broader research and development and outreach strategies?

While we believe our study makes an important contribution to our understanding of sponsored open source communities — and open source collaboration more broadly — important questions remain unanswered. For example, how does the creation of such communities affect firm business models and practices more generally? There are many opportunities for further research on sponsored communities. One area is the long term impact of the tension between sponsor and participant goals — are external participants eventually discouraged by the sponsor's ongoing control, or do sponsors increase accessibility over time as they learn how to do so without surrendering full project control.

There are also questions about the changes of sponsorship over time. The imprinting of an organization at founding is certainly important (cf. Stinchcombe, 1965), but control over the community can change over time. Communities may transition from autonomous to sponsored, as has happened when a community founder (usually a hobbyist-programmer) forms a company to monetize the value of the code. Or they may transition from sponsored to autonomous, often as part of a larger transition from a proprietary software project to a sponsored open source community to an autonomous community. Both Mozilla (now Firefox) and Eclipse went through such transitions, but they have received little empirical examination. What affects the evolution of such projects? What consequences do these changes have for the code, the community and for innovation in general?

Our work is suggestive but hardly definitive on the role of technical aspects of openness. Consistent with Baldwin and Clark (2005), our informants suggested that technical structure (modularity) was one key aspect, while the other (along the lines of software engineering practice) was coding style. But are these categories mutually exhaustive? What about design elements (such as well-documented programming interfaces) that span both categories? How would these be measured? The next step would be to test the predictive value the constructs we identified. Would our social/structural measures predict participation? Or would technical openness have greater predictive value? Or is it some other factor, such as product quality, the size of the target market, or price of the existing alternative?

Finally, as with any study conducted in a single industry context, there are opportunities to verify the generalizability of the findings — in this case, whether the sponsorship processes identified in open source software apply to other types of distributed content generating or innovation-focused online communities. There are a host of peer production and content generation communities that have flourished in recent years – some nonprofit and some commercial, however how a community's participation architecture affects either community or commercial growth has not been teased apart. Our hope is that this framing offers a starting point.

References

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

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

Armstrong, Arthur and John Hagell 1996, "The real value of online communities," *Harvard Business Review* (May), pp. 134-141.

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

Baldwin, Carliss Y. and Kim B. Clark, 2005, "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,* June 2005.

Baldwin, Carliss Y. and Kim B. Clark, 2006, "The Architecture of Participation: Does Code Architecture Mitigate Free Riding in the Open Source Development Model?" *Management Science* 52 (7), 1116 - 1127.

Baldwin, Carliss Y, Siobhan O'Mahony and James Quinn, 2003, "IBM and Linux," Harvard Business School Case 9-903-083.

Bijker, Wiebe E, Thomas P. Hughes and Trevor Pinch (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.

Christensen, Clayton M. and Richard S. Rosenbloom, 1995, "Explaining the attacker's advantage: technological paradigms, organizational dynamics and the value network," *Research Policy* 24 (2), 233-257.

Dahlander, Linus., Magnusson, Mats G., 2005, "Relationships between open source software companies and communities: observations from Nordic firms," *Research Policy,* 34 (4), 481-493.

Dahlander, Linus, Wallin, Martin W., 2006, "A man on the inside: Unlocking Communities as complementary assets," *Research Policy,* 35, (7), 1243-1259.

Edmondson, Amy and E. McManus. 2007. "Methodological Fit in Management Field Research." *Academy of Management Review* (forthcoming).

Fleming Lee and David M. Waguespack, 2007. "Brokerage, Boundary Spanning, and Leadership in Open Innovation Communities," *Organization Science,* 18, (2), 165-180.

Franke, Nikolaus and Shah, Sonali, 2003. "How communities support innovative activities: An exploration of assistance and sharing among end-users," *Research Policy* 32 (1), 157-178.

Franke, Nikolaus and Von Hippel, Eric, 2003, "Satisfying heterogeneous user needs via innovation toolkits: the case of Apache security software," *Research Policy* 32 (7), 1199-1215.

Glaser, Barney G. and Anselm L. Strauss, 1967, *The discovery of grounded theory; strategies for qualitative research,* Chicago: Aldine Publishing.

Godwin, Mike, 1994, "Nine principles for making virtual communities work," *Wired* 2.06.

Hargadon, Andrew B. and Yellowlees Douglas, 2001. "When Innovations Meet Institutions: Edison and the Design of the Electric Light." *Administrative Science Quarterly,* 46 (3), 476-501.

Hargrave,Timothy J. and Van de Ven, Andrew H. 2006, "A Collective Action Model of Institutional Innovation," *Academy of Management Review,* 31 (4), 864-888.

Hars, Alexander and Ou, Shaosong, 2002. "Working for free? Motivations for participating in open-source projects." *International Journal of Electronic Commerce*, 6 (3), 25-39.

Hertel, Buido, Sven Niedner and Stefanie Herrmann, 2003, "Motivation of software developers in open source projects: an Internet-based survey of contributors to the Linux kernel," *Research Policy* 32 (7), 1159-177.

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

"Interview: Guido Van Rossum; Benevolent Dictator for Life" 2005. *Linux Format* 64 (March), pp. 60-63.

Isaak, Jim, 2006. "The Role of Individuals and Social Capital in POSIX Standardization," *International Journal of IT Standards and Standardization Research* 4 (1), 1-23.

Jeppesen, Lars Bo and Lars Frederiksen, 2006, "Why Do Users Contribute to Firm-Hosted User Communities? The Case of Computer-Controlled Music Instruments," *Organization Science*, 17 (1), 45-63.

Lakhani, Karim R. and Eric von Hippel (2003) "How Open Source Software Works: "Free" User-to-User Assistance," *Research Policy* 32 (6), 923-943.

Lakhani, Karim and 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. and 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, and Jean Tirole, 2002. “Some Simple Economics of Open Source,” *Journal of Industrial Economics,* 52 (2), 197-234.

Lerner, Josh, and Jean Tirole, 2005, “The Scope of Open Source Licensing,” *Journal of Law, Economics, and Organization,* 21 (1), 20-56.

MacCormack, Alan and Kerry Herman, 1999, “Red Hat and the Linux revolution,” Harvard Business School Case 9-600-009.

MacCormack, Alan, John Rusnak, and Carliss Y. Baldwin (2006) "Exploring the Structure of Complex Software Designs: An Empirical Study of Open Source and Proprietary Code." *Management Science* 52 (7), 1015 - 1030.

Markus, M. Lynne, 2007. “The Governance of Free/Open Source Software Projects: Monolithic, Multidimensional, or Configurational?” *Journal of Management and Governance, Journal of Management and Governance,* 11 (2), 151-163.

Mowery, David C. and Timothy Simcoe. 2005. “Public and Private Participation in the Development of and Governance of the Internet.” In Richard R. Nelson, ed. *The Limits of Market Organization*. New York: Russell Sage.

Murray, Fiona and Siobhan O’Mahony. 2007. “Exploring the Foundations of Cumulative Innovation: Implications for Organization Science”, *Organization Science* 18 (2), 1006-1021.

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 Joseph Feller, Brian Fitzgerald, Scott A. Hissam and Karim R. Lakhani, eds, *Perspectives on Free and Open Source Software*. Cambridge, Mass: MIT Press, pp. 393-413.

O’Mahony, Siobhán, 2007. “The governance of open source initiatives: What does it mean to be community managed?” *Journal of Management and Governance,* 11 (2), 139-150.

O’Mahony, Siobhán and Ferraro, Fabrizio. 2007. “The emergence of governance in an open source community”, *Academy of Management Journal* 50 (5), 1079-1106.

O’Reilly, Tim. 2005. “The open source paradigm shift”, In Joseph Feller, Brian Fitzgerald, Scott A. Hissam and Karim R. Lakhani, eds, *Perspectives on Free and Open Source Software*. Cambridge, Mass: MIT Press, pp. 461-482.

Prügl, Reinhard and Martin Schreier, 2006, “Learning from leading-edge customers at The Sims: opening up the innovation process using toolkits,” *R&D Management,* 36 (3), 237-250.

Raymond, Eric. 1999. *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary,* Sebastopol, Calif.: O’Reilly.

Robbins, Jason, 2005. “Adopting Open Source Software Engineering (OSSE) Practices by Adopting OSSE Tools,” in Joseph Feller, Brian Fitzgerald, Scott Hissam and Karim Lakhani, eds., *Perspectives on Free and Open Source Software*. Cambridge, Mass.: MIT Press, pp. 245-264.

Rosen, Lawrence, 2004, *Open Source Licensing: Software Freedom and Intellectual Property Law,* Upper Saddle River, NJ.

Rosenkopf, Lori and 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, pp. 403-424.

Rosenkopf, Lori and Tushman, Michael L, 1998. “The Co-evolution of Community Networks and Technology: Lessons from the flight simulation industry,” *Industrial and Corporate Change* 7 (2), 311-346.

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

Sabel, Charles F**.** 1984. “Industrial Reorganization and Social Democracy in Austria,” *Industrial Relations* 23 (3), 344-362.

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

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

Shah, Sonali, 2006. “Motivation, Governance, and the Viability of Hybrid Forms in Open Source Software Development,” *Management Science,* 52 (7), 1000-1014.

Stinchcombe, Arthur 1965. “Social Structure and Organizations,” In James G. March, ed., *Handbook of Organizations*. Chicago: Rand McNally.

Strauss, Anselm L. and Juliet Corbin, 1990. *Basics of qualitative research: grounded theory procedures and techniques*. Newbury Park, Calif.: Sage Publications.

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

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 (1-2), 23-51.

van de Ven, Andrew, and Raghu Garud, 1989, “A framework for understanding the emergence of new industries,” *Research on Technological Innovation, Management and Policy* 4, pp. 195-225.

von Hippel, Eric, 1988. *The Sources of Innovation,* New York: Oxford University Press.

von Hippel. Eric, 2001. “Innovation by User Communities: Learning from Open-Source Software,” *MIT Sloan Management Review* 42 (4), 82-86.

von Hippel, Eric, 2005 *Democratizing Innovation,* Cambridge, Mass.: MIT Press.

von Hippel, Eric and Georg von Krogh, 2003, “Open source software and the ‘private-collective’ innovation model: Issues for organization science,” *Organization Science* 14 (2), 209-223.

von Krogh, Georg, Spaeth, Sebastian and 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, 2007. “Value Capture and Value Networks in Open Source Vendor Strategies,” Proceedings of the 40th Annual Hawai‘i International Conference on System Sciences, Waikoloa, Hawai‘i, p. 176.

West, Joel and Jason Dedrick, 2001, “Open Source Standardization: The Rise of Linux in the Network Era,” *Knowledge, Technology & Policy*14 (2), 88-112.

West, Joel and Scott Gallagher, 2006, “Challenges of open innovation: the paradox of firm investment in open-source software,” *R&D Management,* 36 (3), 319-331.

West, Joel and Siobhán O’Mahony, 2005. “Contrasting Community Building in Sponsored and Community Founded Open Source Projects,” Proceedings of the 38th Annual Hawai’i International Conference on System Sciences, Waikoloa, Hawaii, p. 196c.

West, Joel, Wim Vanhaverbeke and Henry Chesbrough, 2006, “Open Innovation: A Research Agenda,” in Henry Chesbrough, Wim Vanhaverbeke, and Joel West, eds, *Open Innovation: Researching a New Paradigm*. Oxford: Oxford University Press, pp. 285-307.

Williams, Ruth L. and Joseph Cothrel, 2000, “Four Smart Ways to Run Online Communities,” *Sloan Management Review* 41 (4), 81-91.

Tables and Figures

Founding Date	Project	Type of Software	Sponsor	Sponsor type	Interviews
1983	Sendmail	electronic mail server	Sendmail	Open source startup†	1
1990	Berkeley DB	database	Sleepycat	Open source startup†	2
1995	MySql	relational database	MySQL AB	Open source startup	1
1997	PHP	web scripting language	Zend	Open source startup	2
1998	Mozilla	Web browser	Netscape	Proprietary I.T. firm	3
1998	Jikes	Java compiler	IBM	Proprietary I.T. firm	2
1999	Darwin	operating system kernel	Apple	Proprietary I.T. firm	3
2000	OpenOffice	Office productivity suite	Sun Microsystems	Proprietary I.T. firm	4
2001	Eclipse	IDE/application platform	IBM	Proprietary I.T. firm	6
2002	Helix	media streaming	RealNetworks	Proprietary I.T. firm	2
2003	Chandler	information manager	Open Source Applications Foundation	Nonprofit corporation	3
2004	Sugar	customer management	SugarCRM	Open source startup	1
				Number of interviews	29

† Originally university sponsored

Table 1: Sample of Sponsor Founded Open Source Communities

		Form of Openness		*Proprietary Model*
		Transparency	**Accessibility**	
Dimension of Participation Architecture	**Production –** the way that the community conducts production processes	Ability to read code and observe or follow production processes	Ability to change code directly	Production remains within a single corporation
	Governance – the processes by which decisions are made within the community	Publicly visible governance, observers can understand how decisions are made	Ability to participate in governance	The corporation makes all decisions at its own discretion
	Intellectual Property – The allocation of rights to use the community's output	Rights to use code and access source code	Ability to reuse and recombine code in the creation of derivative code	Limited use rights are granted by the corporation for a licensing fee

Table 2: Mapping forms of openness against dimensions of open source

Design Parameter	*Production* Live Code Access	*Production* Public Commit Process	*Production* Subproject Creation	*Governance* Nonprofit Foundation	*Governance* Membership	*Governance* Member Fee	*Governance* Community Release Authority	*Intellectual Property* Code Owned by Foundation	*Intellectual Property* Subproject Ownership	*Intellectual Property* Software License	License Type
<u>*Autonomous*</u>											
Apache	X	X	X	X	individ	no	X	X	foundation	Apache	permissive
Gnome	X	X	X	X	individ	no	X	X	foundation	GPL	reciprocal
Debian	X	X	–	X	individ	no	X	X	–	GPL	reciprocal
Linux	X	*	X	*	–	–	*	*	community	GPL	reciprocal
Linux Standard Base	X	X	X	X	firm & individ	X	X	X	foundation	GPL	reciprocal
<u>*Sponsored*</u>											
Eclipse¶	X	X	X	X	firm & individ	X	X	X	foundation	Eclipse	partly reciprocal
Mozilla¶	X	X	X	X	planned	–	–	X	foundation	Mozilla, GPL	partly reciprocal
OpenOffice	X	X	X	planned	planned	–	–	–	sponsor	LGPL	partly reciprocal
Helix	X	X	X	–	–	–	–	–	sponsor	RPSL, GPL	reciprocal
Sugar	X	–	X	–	–	–	–	–	participants	Sugar*	partly reciprocal
PHP	X	X	–	–	–	–	–	–	–	PHP	permissive
Chandler	X	–	–	*	–	–	–	–	–	GPL	dual
Jikes	X	–	–	–	–	–	X	–	–	CPL	partly reciprocal
Darwin	X	*	–	–	–	–	–	–	–	APSL	permissive
MySql	X	–	–	–	–	–	–	–	–	GPL	dual
Sendmail	–	–	–	–	–	–	–	–	–	Sendmail *	dual
Berkeley DB	–	–	–	–	–	–	–	–	–	Sleepycat	dual

Communities are listed in order of decreasing level of overall openness. ¶ Evolved from sponsored to autonomous.
*Notes on specific communities:

- Linux: allows external committers but the process is not public; has an affiliated nonprofit that influences but does not control the project; release authority is vested in key individuals; code in the Linux kernel is licensed by contributors to the community but none is owned by the community
- Chandler was founded by a nonprofit
- Darwin: outside commit rights allowed on parallel, experimental code repository
- Sendmail, Sugar: License is not approved by the Open Source Initiative

Table 3: Design Parameters for Autonomous and Sponsored Open Source Communities