Chapter 18:

Managing the Boundaries of an ‘Open’ Project

Fabrizio Ferraro
IESE Business School
fferraro@iese.edu

Siobhán O’Mahony
Boston University School of Management
somahony@bu.edu

In The Emergence of Organizations and Markets,
edited by John Padgett and Walter Powell, forthcoming.

December 14, 2009

This research was supported by Stanford University Center for Work, Technology, and Organization, the Social Science Research Council Program on the Corporation as a Social Institution, Harvard University Business School Division of Research. We thank the attendees of the Santa Fe Workshop on the Network Construction of Markets and the Schenna Workshop on the Emergence of Organizations and Markets for their helpful comments, in particular the helpful guidance of Woody Powell, John Padgett, Steve Barley, Lee Fleming, Jeannette Colyvas, Kelley Packalen and Ben Adida. The data collection and processing efforts of John Sheridan, Vikram Vijayaraghavan and Jordi Torrents were much appreciated. All errors or omissions are our own. Neither author has any financial interest in Linux or open source companies.
Abstract

Theorists have speculated how collectively managed projects like Wikipedia and Linux manage the production of complex knowledge goods in an open and public environment, but little research has examined how such projects manage their boundaries with their environment. Our research uses a multi-method approach to understand how one open source software project, Debian, developed a membership process to establish boundaries to an open and public project. We examined the project’s face-to-face social network during a five-year period (1997-2002) to determine how changes in the social structure affected the determination of gatekeepers and of membership boundaries. While the amount and importance of a contributor’s work increased the probability of becoming a gatekeeper, those more central to the projects’ social network were more likely to become gatekeepers and influence the membership process. A greater understanding of the mechanisms projects use to manage their boundaries has critical implications for collectively managed projects operating in open and distributed environments. Our findings also help explain how the evolution of network structures coevolves with the construction of new social orders.
INTRODUCTION

Both scholars and observers of the rise of Wikipedia and open source software projects such as Linux often wonder how collectively managed projects that are open to any and all manage the production of complex knowledge goods. If the boundaries of public collective projects are open to any volunteer, how can the quality of complex knowledge goods be sustained? Without the credentialing of knowledge, appropriate organizational controls and adequate financial incentives surely such settings are ripe for malfeasance, cooptation or just inferior work products. Yet, empirical studies demonstrate modest discrepancies between encyclopedia texts produced by credentialed experts and those of Wikipedia’s motivated body of volunteers (Giles, 2006) and industries continue to clamor over the “open” approach to innovation (Chesborough, 2005; 2006). Furthermore, since these two collective projects have become world renowned, the public and open approach to producing other types of goods has become increasingly popular (e.g. von Hippel, 2005; Jespperson&Lakhani, forthcoming).

Despite the fact that determining an organization’s boundaries with the environment is fundamental to economic (Williamson, 1975, 1981); sociological (Lamont & Molnar, 2002); and organizational theories (Santos &Eisenhardt, 2005), little research has examined how seemingly open projects create or manage organizational boundaries. Do collectively managed projects remain open to all or do they funnel and narrow over time – becoming increasingly closed to a select population of elite incumbents? If project boundaries remain open, how do they ensure that incoming contributors do not violate the project’s mission? If project boundaries close, who become the gatekeepers of such forms? Our research draws upon both field and network methods to explore how social networks affect the emergence of organizational boundaries in novel collective forms – focusing on membership
as a core boundary of interest. With qualitative data, we unpack the threats faced by open projects and the membership mechanisms designed to manage these threats. With quantitative data, we analyze the structure of the project’s network to predict who designs the new membership process, thereby becoming gatekeepers to the project. In doing so, we explain how networks and social structures co-evolve to affect organizational boundaries in novel forms.

**Managing Boundaries in Knowledge Communities**

Although the examination of organizational boundaries was once dominated by transaction cost economics (Williamson 19785; 1981), organizational theorists are starting to consider power, competence, and identity as under appreciated and critical elements to providing a holistic view of organizations and their environment (Santos & Eisenhardt, 2005). For example, in their research comparing the Burning Man and open source communities, Chen and O’Mahony (2009) identified two boundaries that were critical to the creation of new organizations: the boundary between individuals and the organization (determining what the organization could and could not do) and the boundary between the organization and the market (determining the degree to which community efforts could be commoditized). Santos and Eisenhardt might argue that the first boundary is one of power, demarcating an organization’s domain of influence while the second boundary is one of identity, enabling sensemaking among organizational members and inspiring attachment. Establishing both boundaries proved to be important to the ability of the Burning Man and open source communities to represent the interests of their growing membership base and sustain the motivation of volunteer contributors (Chen and O’Mahony, 2009).
To advance theory, Santos and Eisenhardt urge scholars to move beyond efficiency arguments such as make or buy decisions, to “problem-driven boundary phenomena” that are less atomistic in nature. New boundary choices in nontraditional settings or boundary decisions in nascent industries and organizations are likely to generate much needed new thinking in this area (Santos and Eisenhardt, 2005). One example of a problem-driven boundary decision is deciding who can legitimately participate, contribute and join in an organization’s activities. When organizations establish the criteria for recruiting and selecting new members, they are simultaneously selecting the talents and values that will permeate the organization and who will gain a say on guiding the organization’s mission. Determining membership is a critical function by which organizations can sustain themselves when facing any type of natural attrition. This is especially true in professional and knowledge producing communities where contributors to knowledge projects are also trying to exert control over their work through the setting of boundaries.

For example, the literature on the professions shows how professions collectively determine criteria to keep “unqualified” members out of the profession and lay claim to a specific knowledge base (Sarfatti-Larson, 1979). Boundaries help distinguish layman from experts, science from non-science (Lamont and Molnar, 2002). Scientists engage in “boundary-work” to distinguish “true science” from “non-science” and thus retain authority over their expertise (Gieryn, 1983; 1999). This is particularly the case when scientists strive to distance their work from political or commercial aims (Moore, 1996; Guston, 2000).

To develop more generalizable theories on boundary processes that may be common across apparently unrelated phenomena, Lamont and Molnar (2002: 168) suggest that scholars “undertake the systematic cataloging of the key mechanisms associated with the activation, maintenance, transposition or the dispute, bridging, crossing and dissolution of
boundaries” (2002: 187). Boundary disputes in particular offer a critical empirical entrance point to understanding processes of boundary activation and revision. A boundary dispute may question the blurring, penetration or division of two previously distinct social worlds. To inform our present inquiry, we identified two critical boundary disputes in science and knowledge producing communities: 1) managing the boundaries between open and commercial science and 2) managing the boundaries between open source software and industry.

Open Science and Commercial Science. Many scholars have become concerned that the boundaries between public (or ‘open’ and academic) and commercial science have become blurred to detrimental effects (Dasgupta and David, 1994). The open and commercial science communities are interdependent in the creation of new knowledge but diverge in their reward systems and in their dissemination and use of that knowledge (Dasgupta and David, 1994). The logic of open science assumes that the production of science is a public endeavor (David, 2000; 2003) and encourages universalistic standards based on competence or merit (Merton, 1973). The practice of open science demands full and timely disclosure of methods and findings in order to allow scientists to replicate and verify each other’s work (Merton 1973; Latour&Woolgar, 1979). While the goal of commercial science is to increase the stream of rents that can accrue from rights to private knowledge, the goal of open science is to add to the stock of public knowledge (Dasgupta and David, 1994).

Instead of property rights, scientists are granted priority for discoveries made. This provides scientists with an incentive to share their discoveries early, helping scientists to avoid duplication and advance the field more rapidly. David argues that the institutional framework supporting public science can be undermined if too great an emphasis is placed on property right protections (2000). Thus, many scholars have examined how the Bayh
Dole Act, which permits universities to take an expanded role in licensing academic research for commercial purposes, has affected the funding, conduct, use and dissemination of university research (Owen-Smith, 2003; Owen-Smith and Powell, 2003; Mowery and Sampat, 2001; 2004; Mowery and Ziedonis, 2002). Of concern is whether the norms of open science are compromised when university endeavors cross commercial boundaries.

There is suggestive evidence that the blurring of commercial and university science boundaries has, in the last twenty years, had some effects (Owen-Smith, 2003; Owen-Smith and Powell, 2003; Huang and Murray, 2009). Universities well connected to industry had patent portfolios with greater impact, but these relationships reached a point of diminishing returns. Technology licensing offices that were too closely tied to industry had less innovative patent portfolios (Owen-Smith and Powell, 2003). The suggestion that university research has become more applied due to close relationships with commercial entities highlights both the importance and the fragility of managing boundaries between public and private organizations as these relationships can actually affect the type of knowledge produced. Most scholars agree that commercial support of university research is an imperative as public support for it has declined (Mowery and Sampat, 2004). The question is how a more integrated commercial and academic relationship can prosper without unduly influencing the direction of open science. Sustaining openness and pluralism without risking cooptation from commercial entities is a central concern. This is also a boundary dispute to which open source software projects are starting to gain exposure.

Open Source Software and Commercial Software. The production of open source software is often compared to the ‘open science’ process of peer review (Dalle and David, 2003; Kogut and Meitu, 2002; Raymond, 1999) where work and method are critically evaluated by peers with informed skepticism (Merton, 1973; Latour&Woolgar, 1979). This comparison is only
partially true – for example, academia has a long history indoctrinating graduate students to the norms of science that open source projects do not share. Open source software projects do embrace open science principles and are guided by powerful norms that reinforce or discourage certain types of behavior. O’Mahony (2007) defines community-managed open source software projects as embracing principles of: (1) independence, (2) pluralism, (3) representation, (4) decentralized decision-making, and (5) autonomous participation. Most importantly, project organization and decision-making occurs in public forums independent of any one firm and is not influenced by authority relations that stem from employment (O’Mahony, 2007). However, these community projects lack the socialization, institutional and professional structures academe provides to guide training and access to the production of new knowledge.

Both open science and open source software are similar in the challenge they face with respect to boundary disputes between the open and public production of knowledge and the commercial capturing of value. Since the term “open source” was created in 1998, open source software has attracted a more diverse group of supporters interested in commercializing open source software despite the fact that it is produced in public forums and distributed with open source licenses. Some open source software projects have been receptive to a commercial audience and engaged in synergistic relations with firms, but remain wary that their culture, practice, and code may be compromised. Sponsorship from industry dominants has introduced a new challenge: how to maintain independence and neutrality in the face of industry support (O’Mahony, 2002; O’Mahony and Bechky, 2008).

While open communities producing new knowledge benefit from the diversity of their contributors, contributors must share a common goal in the project’s success as a non-commercial entity for the project to sustain itself. In a completely open environment, not all
contributors may have the project’s best interests in mind. As the number of contributors to open source projects grows, so does the diversity of contributor skill and motivation. The potential for someone to co-opt or hijack a project or unwittingly introduce code owned by someone else looms large. Good intentions without skill can be equally dangerous. For quality in this context is not merely technical but also legal, referring to code that is free of proprietary licenses. Open source projects do not want to unknowingly accept code that might conflict with open source licenses. Thus, project members want to ensure that potential contributors share the project’s values and do not introduce code that might jeopardize the project’s boundary with proprietary software. In this sense, managing project membership affects not only the boundary of the project itself, but also helps preserve the project’s legal boundaries with the commercial world.

Recent research has attended to ways in which open projects are becoming more bounded. For example, despite the popular belief that open source contributors give their work away, many contributors to open source software projects now assign copyrights to a non-profit foundation designed to hold the group’s efforts in trust (O’Mahony, 2003). With growth in the scale of code, contributors, and industry sponsors, several open source projects have sought to make clearer determinations of membership and rights (von Krogh, Spaeth, and Lakhani, 2003; Michlmayr and Hitt, 2003). In their study of the FreeNet project, von Krogh and colleagues (2003) discovered that potential contributors with particular ‘joining scripts’ and contributions of code were more likely to be awarded developer status. O’Mahony and Ferraro (2007) showed how one community’s conception of leadership changed over time as the project scaled and became unwieldy. Thus, the question is how do open source communities manage the boundaries of an open project? Powell (1990) predicted that network forms would face novel problems of control and that membership in
a community would require new organizational practices (Powell et al, 1996: 142). Our research explores how one project managed these exact challenges by examining the design of their new membership system. We then examine who became the gatekeepers of this system – revealing how the project’s social network and organizing structure co-evolved over time.

METHODS

Research Setting

Despite the fact that the distributed setting of an open source project implies the existence of powerful social networks, a social network approach has yet to be used to explain the evolution of a project’s social structure. With unique longitudinal network data, we examined the evolution of the Debian Linux project’s social network over a five-year period (1997 – 2001) to assess how changes in the network structure affected the design of membership mechanisms. Debian produces the largest and most popular non-commercial Linux operating system distribution and has been in existence for over ten years. Like other commercial distributions such as Red Hat™ (www.redhat.com), Debian integrates the Linux kernel maintained by Linus Torvalds and other kernel hackers (www.kernel.org) with thousands of other software packages to create a complete self-installing distribution. Unlike RedHat, Debian is not a commercial entity and does not sell its code or pay its programmers. Debian has over 1,000 volunteer programmers\(^1\) distributed in over 40 countries who collectively maintain over 8,000 software packages. The vast majority of coding activities are publicly accessible and the software produced can be downloaded for

\(^1\) Some developers engage in wage earning activities that allow them to work on Debian as part of their paid work: they are what we define as sponsored contributors. Others are volunteer. Participation in the project is always voluntary.
free. But, access rights to the code base must be managed so as not to jeopardize the project’s security.

Unlike online communities with fluid boundaries and potentially anonymous, shifting members and identities, the Debian project needed a way to trust the identity of project contributors in order to grant them the right to contribute directly to a project’s code base. Membership could be fluid, but it could not be indeterminate - the allocation of access rights had to be known and distinguishable. Since contributors may never meet each other, they faced a unique problem: how to verify the identities of individuals distributed around the world. Thus, the Debian project began using public key encryption as a way to build trust and authenticate member identities in 1994. This method became, in the spring of 2000, a condition for becoming a project member.

A key is merely a large number that, with the help of a particular cryptographic algorithm, like one offered by “Pretty Good Privacy” (PGP) or GnuPG (GPG), allows text to be encoded and decoded only by the intended recipients. Some cryptography methods, called secret key algorithms, use the same key to encode and decode data. This presents a complicated key distribution problem: how can a distant sender and recipient exchange this secret key without compromising each other’s security?

Public key encryption uses cryptography to solve this problem by using asymmetric keys: a public key encodes the data and a completely different private key decodes the data, allowing a sender and recipient to exchange private information without key distribution. Thus, a user’s private key is never revealed (Network Associates, 1990). Asymmetric cryptography does not, however, solve the problem of certifying a key holder’s identity. Public key cryptography secures the authenticity of the contents of the communication
but not the link between the key and the sender’s identity. To make public-key cryptography useful, a real-world identity must be linked to a given public key.

The Debian project uses ‘key signing’ practices to link individual identity to key ownership. A key is certified when one person digitally signs the public key and user identification packet of another. A key certification is an expression of trust: the signer believes that the public key they sign belongs to the cited person. Some form of identification documentation (usually government issued) demonstrates that a public key belongs to its owner and is represented by the user id packet (Brennen, 2003). This certification does not provide assurance as to the authenticity of their identification documents, but provides assurance that a particular identity is assigned to a particular key (Network Associates, 1990).

In a globally distributed environment like Debian, where everyone cannot meet everyone else, responsibility for validating public keys is delegated to trusted others. Key signers are explicitly encouraged to consider not only their own security requirements but, the interests of others who may rely on their judgment (Free Software Foundation, 1999).

“Key signing has two main purposes: it permits you to detect tampering on your keyring, and it allows you to certify that a key truly belongs to the person named by a user ID on the key. Key signatures are also used in a scheme known as the web of trust to extend certification to keys not directly signed by you but signed by others you trust” (Free Software Foundation, 1999: 13).”

Certificates provide validation, but people are trusted to be judicious when validating the certificates of others. A ‘web of trust’ is a collection of key signings that allows people to rely upon third party verification of other’s public keys. The web of trust assumes that the more people who have signed each other’s key (the greater the density of the network), the more reliable is the information authenticated. “The more deep and tightly inter-linked the
web of trust is, the more difficult it is to defeat” (Brennen, 2003). There is no limit to the
number of people that can sign a key.

Debian contributors have their keys signed by hosting or attending ‘key signing
districts’: get-togethers for the purpose of signing each other’s keys. At a key-signing party, a
small group of individuals will bring a copy of their public key and valid photo identification,
meet and certify each others’ public keys. After a key is signed it can then be placed on a
central key server that may be maintained by a ‘keyring coordinator’. Key signing parties are
viewed as critical to enhancing the web of trust, to teaching people about the benefits of
cryptography, and to building technical communities (Brennan, 2003).

[Please don’t sign keys of people you did not personally identify. If you don’t take
this process seriously, you are a weak link in the Web of Trust. If I see that you
signed the key of someone who wasn’t at the event, I won’t sign your key, and I’ll
suggest that others don’t either (Key Signing Party Organizer, July 8, 2001).

As this party organizer explains, violation of key signing protocols can lead to sanctioning
and possible estrangement by other project members. Signing someone’s key without
physical verification of his or her identity breaches the norms of the community and
threatens the web of trust. If someone is viewed as lax in their security requirements, their
ability to maintain the respect of their peers will be compromised.

Since each key signing is dated and requires a face to face meeting, these data
indicate when individual project members met each other and provide a unique longitudinal
network data set. We analyzed these data to determine if an individual’s structural position in
the network affected the attainment of gatekeeper positions. With qualitative data on the
project’s evolution, we examined how the Debian project managed a membership crisis
where an influx of contributors new to the project’s norms, methods, and values began to
overwhelm the project. Together, these two empirical approaches show how organizational
design and social network dynamics are intertwined, and that the former cannot be ignored when attempting to understand the emergence of new social orders.

**Data Collection and Measures**

The data we collected from the Debian keyring consist of gpg and pgp keys signed by dyads between 1994 and 2002. The keyring network was only minimally active during the project’s first two years (1994-1996). Thus, we begin our analysis at the beginning of 1997 when key signing started to become more widely adopted. Table 1 reports the number of developers in the keyring, the rate of growth of the nodes in the network, the number of ties between members, average degree (number of keys signed or people met), standard deviation of degree, number of components, and density of the network from 1997 to 2002.

**Statistical Model.** From the project developer database, we identified the continent of residence for each developer and leadership positions, if any, held over time. In Table 2, we summarize data on the continent of residence with three dummy variables (Europe, North America, and Other) as well as other descriptive statistics such as packages maintained, postings to the mail list, tenure and degree centrality. As a measure of each developer’s contribution to the project, we collected data from the project’s bug tracking database on the number of software packages each developer maintained in 2001 and 2002 (the only years available). In both years, developers maintained an average of 7 packages (with a standard deviation of 9 in 2001 and 10 in 2002). Similar to prior studies of the Apache, FreeNet and GNOME projects, a small fraction of maintainers contribute the majority of the work (von Krogh et al, 2003; Mockus, Fielding and Herbsleb, 2000; 2002; Koch and Schneider, 2000). Under 8% of maintainers managed more than 20 packages in 2001 and 2002. The maximum number of packages maintained by any one person was 81 in 2001 and 101 in 2002.
Since the number of packages maintained provided only a raw measure of quantity of effort, we created a measure of the criticality of a developer’s work on the project by computing the package popularity. Since early 2003, Debian users could install a “popularity-contest package” that automatically calculates the number of people that use a particular package regularly. We computed the raw sum of the votes for all packages maintained by individual developers to measure the criticality of their work to others. We also included a variable to measure other forms of participation that did not involve direct coding by measuring the number of mailing list postings each developer contributed to the project’s primary mailing list focused on development: “debiandevel”.

For each year we computed a measure of developer project tenure, counting the months since they first signed a key. We used the keyring data to measure the degree centrality of each developer, which is the number of other developers each one of them has met face-to-face at least once. Finally, we created a variable to measure ties to the project leader indicating which developers signed the project leader’s key. We wanted to control for any effect that the project leader might have on the final composition of the new maintainer committee: our dependent variable of interest. As described in more detail in the next section, the New Maintainer Committee was established to design the new membership process and the rules that would admit new entrants to the project. This group in essence becomes the project gatekeepers once new boundaries are established.

To show how the structural position of the developers affected the composition of the New Maintainer Committee, we estimated a logistic regression model (Long, 1997), testing whether degree centrality in the network (1997 – 2000) affected the composition of

---

2 We also computed betweenness centrality, which measures the extent to which an actor can broker communication between other actors (Freeman, 1979; Marsden, 1982; Wasserman and Faust, 1994) but since this measure was highly correlated with degree centrality, we only used the latter in our analysis.
the new maintainer committee in 2001-2002, after controlling for the level and criticality of contribution, ties to the project leader, tenure in the project and geographic location.

*Field Research.* In order to understand these data in the context of the project’s evolution, 76 informants from the open source community at large were interviewed, six of them in leadership positions within Debian. Online documentation such as mailing list archives, meeting notes, and other formal project documents offered an additional source of data. With these data, we analyzed the project's evolution along six critical phases: 1) project initiation; 2) success and new vulnerabilities; 3) membership crisis; 4) designing the membership process; 5) emergence of gatekeepers; and 6) narrowing the pipeline. We analyzed how project members’ conceptualization of membership evolved over time and how this affected the project’s boundary with the very open and public environment in which the project operated.

**FINDINGS**

**Designing a Membership Process**

*Project Initiation.* On August 16, 1993, the founder of Debian proposed developing an easily installable packaged version of the GNU³/Linux operating system to a Usenet newsgroup. He wanted to create a complete operating system that would be ‘commercial grade’ but not commercial, and be managed differently from the Linux kernel project.

Rather than being developed by one isolated individual or group, as other distributions of Linux have been in the past,⁴ Debian is being developed openly in the

---

³GNU is a recursive acronym that represents the phrase “GNU is Not Unix”. The GNU system developed by Richard Stallman was designed in opposition to the proprietary restrictions associated with UNIX.

⁴This reference to other Linux distributions managed by one person likely refers to the Linux kernel managed by Linus Torvalds.
spirit of Linux and GNU. The primary purpose of the Debian project is to finally create a distribution that lives up to the Linux name [...]. It is also an attempt to create a non-commercial distribution that will be able to effectively compete in the commercial market (Murdock, 1994).

About two-dozen people responded to the posting and the founder created a new mailing list specific for this project named “Debian.” Between 1993 and 1996, the founder, with the help of Usenet respondents, collectively designed a modular package management system.

A package is a unit of code that can be maintained independently from the rest of the operating system but has a standardized interface that allows integration with other packages. To maintain a package is to manage the receipt and review of code contributions from other contributors (called ‘upstream maintainers’) and ‘package’ these smaller contributions into a discrete module. A modular package system enables many people who are not physically co-located to contribute to the project by permitting different development activities to be conducted in parallel. From 1994 to 1995, the Free Software Foundation supported the founder to design a technical infrastructure that could handle a large number of contributors. The first whole number release (1.1), announced in June of 1996, had 474 packages.

In the months leading up to the project’s first official release in July of 1997, members debated how to manage the project’s status as a non-commercial entity. Five issues were critical to establishing boundaries with the business world: 1) garnering legitimacy as a non-commercial entity; 2) determining how to logistically distribute their software; 3) raising funds to support the project’s legal expenses as a non-profit; 4) distinguishing ‘official’ copies of Debian from versions modified for commercial purposes;

---

5 The origins of the project’s name stems from a combination of the founder and the founder’s wife’s names.
and 5) determining how, if at all, commercial entities should contribute to the project.

Exploration of these concerns challenged the meaning of ‘non-commercial’ as it was initially conceived and reflected competing goals among members: to both control their product and yet disseminate it broadly.

We don’t want to be in the CD manufacturing business, the import-export business, or the order fulfillment business. We want to get Debian into as many people’s hands as we can, for as little money as possible (Posting to Debian Development Mail list, January 17, 1997).

Project members wanted to acquire the legitimacy associated with shrink wrap software, but Debian did not have the capital to manufacture a physical distribution. Commercial involvement would help them establish a larger market share than Internet downloads would permit, but they did not want to sell their work.

One proposal to contract with firms to distribute Debian for two dollars was perceived by others as crossing the ‘non-commercial’ line. Project members questioned whether it was within their charter to ask, mandate, or suggest contributions from firms. The project leader that informally took over the project when the founder resigned at the end of 1996 angrily defended the commercial appeal of the project in the following post.

I AM NOT TRYING TO TURN THE PROJECT INTO A COMMERCIAL ORGANIZATION. IT IS A NON-PROFIT. I WANT TO RAISE OUR PERCEPTION IN THE PUBLIC BY MAKING OUR PRODUCT _LOOK_ COMMERCIAL (Posting to Debian Development Mail list, January 19, 1997, original format).

What constituted a non-commercial distribution was hotly contested and in the end, a consensus agreed that Debian would not sell its code. Individuals and firms could freely download and resell the Debian distribution with no fee. In return, some firms could, at their choice, donate a portion of their proceeds to Debian.
**Success and New Vulnerabilities.** With these agreements in place, the project released the first ‘official’ distribution (1.3) with 974 packages contributed by 200 developers. This event also marked the growth of the keyring network. Although the keyring was initiated in 1994, only 13 project members had signed each other’s keys at the start of 1997. The keyring network grew to 82 people in 1998 and 176 in 1999 (Table 1). This rapid growth may have been stimulated by media coverage that began after the first release (Figure 1), but also reflected concerns over the threat of ‘Trojan’ contributors. A Trojan contributor would be a volunteer or ‘malicious contributor’ who purposively introduced bugs or viruses to the project.

Debian’s technical success in building a complete distribution and its subsequent popularity meant that, like other Linux distributions, it was now a threat to other commercial operating systems, making Debian vulnerable to anti-competitive tactics. However, well-intentioned but unskilled developers could create equally detrimental effects. Debian developers all have the same access rights, and can upload anything into the project’s code archive. This has the potential to affect all other packages. Typically the official maintainer of a package makes such an upload. Changes made by a non-maintainer will not carry the same status as those made by someone listed as the maintainer. If someone fixes a bug in another person’s package, that bug will be tagged and fixed, but the maintainer will have to close the bug in the database themselves, signaling that the person responsible has reviewed the work of the non-maintainer. Newcomers to Debian who were not fully cognizant of Debian procedures could wreak havoc with Debian’s detailed code formats and operating procedures. Members were torn between reconciling the need to welcome people interested in Debian with the need to protect the project from potentially destructive outsiders.

---

6 See Michlmayr and Hill (2003) for more description of the new maintainer upload process.
Mailing list archives indicate that the idea of using a key ring to authenticate contributor identity was proposed by the person who initiated the key ring network in 1994.

I think that at least one of our objectives should be to establish a socio-legal comeback in the case of a malicious developer. This means that we need to verify the real-world identity of the developer somehow. There are several ways to do this, including personal introduction by an existing developer, commercial key-signing, attempting to use PGP web of trust, telephone verification of some kind (Posting to Debian Development mail list, February 28, 1997).

At the time, there was no formal standard membership process and little preliminary screening. As the informant below describes, the ability to articulate areas for contribution was considered evidence of one’s capability to work on the project.

When I applied, I told [the Debian Project Leader], “Here are the packages I want to work on.” Back then it was pretty easy and we didn’t do the identity check at that point. It was pretty easy to assume that if you knew about Debian back then, you were fairly competent and probably understood the basics of what free software was about. It was a new thing back then — free software particularly. (Sponsored Contributor, Former Volunteer, November 9, 2000)

Several mailing list threads discussed ways to secure the identities of contributors, ascertain membership, and determine project decision rights. “There are enough psychos out there to make sure that groups such [as] Debian do not succeed” (Posting to Debian Development Mailing list, October 25, 1997). This discussion persisted for some years.

How could Debian keep the project ‘open’, but ensure that contributors were not only well intentioned, but skilled enough not to inadvertently harm the project? Members were reluctant to articulate a formal set of skill requirements or make too many demands of volunteers. “I’m not sure about competence or integrity requirements [to acquire maintainer status]; somehow it goes against the grain for someone who is not issuing my paycheck” (Posting to Debian Development Mail list February 27, 1997).
The second whole number release (2.0), announced in July 1998, had 1500 packages and over 400 developers. As Figure 1 shows, this release coincides with a sharp increase in media attention devoted to Debian. Shortly after Debian’s fifth birthday, the third leader initiated the collective drafting of a Constitution to outline the roles and rights of the project leader and project members. The Constitution delimits the authority of the Debian leader and bounds the group’s authority over each other. Members have the right to: 1) Make technical or non-technical decisions with regard to their own work; 2) Propose or sponsor draft resolutions; 3) Run as a project leader candidate in elections; and 4) Vote for resolutions and leadership elections. The Constitution details specific privileges for members, but did not articulate how one becomes a member. The question of how to prevent a ‘Trojan’ was left unresolved as Debian’s public presence continued to grow.

Membership Crisis. In August of 1999, several package maintainers who were not yet full developers began complaining of the wait to obtain a developer account. A contributor that proved their ability to maintain a package could become a maintainer, but they did not necessarily become a developer or project member. Only developers had accounts to access the code repository and in order to upload a package directly as the project now required members to sign the package with their key.

If anyone could upload to Debian… [twisted facial expression]. We had to guarantee that it actually comes from a trusted source, and that it hasn’t been changed along the way. That is how what we achieve this - by signing the packages…First of all it shows that it is from a trusted source. It is signed by a key, which is a developer. The other thing is it shows that the package hasn’t been modified. Like during the upload someone could come and change the package perhaps. (Volunteer Contributor, DPL, June 20, 2003)
The Developer Accounts Manager (DAM), authorized to assign new accounts, faced a backlog of people interested in the project and had no real way to ascertain the qualifications of a particular maintainer. A few candidates resigned in frustration.

To avoid losing contributors, some developers began ‘sponsoring’ member candidates by uploading their packages for them and signing them with their keys. Postings to the list and interviews with informants suggest that the delay in accepting new maintainers was partly due to a heavy workload for volunteers, but also due to their concerns that recent entrants to the project were actually hindering the project more than they were helping.

The problem is they [new maintainers] are not contributing. And some of them are contributing bugs. They are actually adding bad packages. And there was a lot of kicking from old developers about the new people and how they are not reading anything and doing things, it is always old versus new. So we closed being a new maintainer for while because there were so many new packages coming in that they were not helping with anybody else’s packaging, they were just uploading their own. So what we were seeing is 1,000 new packages a year and that many more bugs per package showing up. But no change right? It was not getting better, it was getting worse. So we closed it off for a year. Then we sat down and wrote up how we wanted it to work (Sponsored Contributor, March 20, 2001).

After doubling in size, project members were frustrated by the growth in bugs relative to contributions from new members, and in particular, the age of bugs unaddressed. New maintainers wanted to work on areas of their interest, not debug existing bugs.

To address the problem, the fourth project leader (newly elected by the powers of the Constitution) made a controversial decision: he closed the project to new maintainers until a new membership process could be designed. Membership would remain closed until April the following year: almost six months. As Figure 1 shows, Debian’s contributor and package growth plateaus.

Debian’s new maintainer team is currently not processing requests. The team wanted to resolve some problems they observed with the way Debianmaintainership is
currently handled, and decided to close new-maintainer until these have been fixed. We are currently working on a new structure for handling new-maintainer requests, and hope to have this finished as soon as possible (Debian Project Leader, Posting to Mail list October 11, 1999).

This announcement was followed by a recruiting call outlining criteria for a new membership committee: the New Maintainer Committee (NMC). Developers were invited to email the leader (privately) and were told that committee members would be selected according to the following criteria:

[T]he following guidelines will be used in selecting new members to the new-maintainer team:
- needs to have a *strong* opinion for free software
- he needs to be able+willing to make long distance phone calls
- He needs to know what he’s doing, that new people need some guidance, we have to prevent ourselves from trojans etc.
- we need to trust him - more than we trust *any* other active person
- He *has to* understand that new-maintainer is *more* than just creating dumb accounts on n machines (New Maintainer Proposal, October 19, 1999)

The need to “trust committee members more than any other active person” suggests that the project leader understood that this committee would become future gatekeepers for the project and wanted confidence in their philosophical commitment above and beyond their effort on the project.

Designing the Membership Process. While there had always been some identification process before granting new developer accounts, it was not standardized. The committee initially proposed a four-stage process: initial contact (with possible phone interview), checking identification, internship, and acceptance. Identity authentication would ensure that members “know that the person actually exists as the person that they say they are, that there is a known location for that person where they can be spoken to” (New Maintainer
Project members engaged in this debate recognized that face-to-face meetings could help instill greater collegiality, trust and respect among members.

Maybe the "meet a developer approach" combined with a brief phone interview is better than a lengthy call from some faceless developer. Plus it gives new maintainers an opportunity to have their key signed, which helps build our web of trust, and the personal contact might socialize against flamemongering (Posting to Debian Project Mail list, October 18, 1999).

The ‘internship’ would allow applicants to prove themselves by testing the candidates’ technical competence, knowledge of organizational procedures, collegiality and commitment, and philosophical agreement with the principles of the project.

It [the internship] allows us a good method to help a new maintainer with his new work and teach him about the Debian system (both technical and organizational). It allows us to get to know the person: is he responsive to bug reports or other requests, is he able to produce a quality product, and also very important: does he agree with our philosophy? (New Maintainer Proposal, October 19, 1999).

Agreement with the project philosophy was critical to ensuring that people did not introduce code licensed under legal terms that differed from the project’s open source license. This proposal led to a discussion of what it meant to be an ‘open project’. Some wondered whether these requirements were too onerous for an open project that valued freedom.

What are the reasons for ever not letting new maintainers in? There are none, I agree. I’m very disappointed that [Debian Project Leader #4] has failed to reopen New Maintainer. This is the biggest failure of his tenure thus far, IMHO [In my humble opinion] (Posting to Debian Project Mail list, December 29, 1999).

Ready and willing would-be contributors posted their frustrations with the closed process and the lack of clear criteria for membership.

My understanding is that the addition of new maintainers is not merely slow, but has been officially stopped. Why? What IS the motivation? Here I am, a highly competent person, a happy and satisfied Debian user, and someone who thinks it's
my duty to contribute back to Debian with some of my labor and talent (*Debian Project Mail List Posting, December 20, 1999*).

However, the new maintainer process did not reopen before the year’s end.

The final membership process adopted required not only identity verification through face-to-face exchange of keys, but sponsorship by an existing member, demonstrated understanding of the community’s philosophy and procedures; demonstrated technical capability; and a written recommendation from an Application Manager. The first new Debian member formally admitted since October, 1999 was admitted in April, 2000. Members of the NMC worked diligently throughout the spring of 2000 to get through the backlog of applicants. By November 10, 2000, 100 people had passed the new maintainer process and several hundred were in progress.

*Emergence of Gatekeepers.* The New Maintainer Committee (NMC) effectively modified the future structure of the project by developing a process that would regulate the flow of new members: in effect becoming architectures of the future network. We used statistical analysis to examine how the structure of the network affected who became gatekeepers of the project. Table 5 presents the logit coefficients for models predicting membership on the new maintainer committee based on the number and popularity of packages maintained, number of mail list postings, tenure, geographic location and the number of ties (degree centrality) of developers in the network in 2001 and 2002\(^7\). We also controlled for ties to the project leader who had final approval authority of the committee.

In the base model, which does not include degree centrality or mail list postings, only the number of packages maintained and ties to the project leader had a significant positive effect on the likelihood of joining the NMC in 2001, while tenure had a negative effect. In the second model for 2001, we included the number of mail list postings made the year prior

\(^7\) The correlation coefficients of the variables are reported, for 2001 and 2002, in tables 3 and 4.
and found little effect. In the third model for 2001, we included degree centrality and found a significant positive effect that is larger than any other variable. Controlling for ties to the project leader, developers with greater degree centrality were much more likely to become a member of the NMC in 2001. The number and popularity of packages maintained continue to have a positive effect, while tenure continues to have a negative effect.

Our findings show that technical contributions are predictive up to a point, but degree centrality has a significant and positive influence on becoming a project gatekeeper. These findings are consistent with the findings of prior research on the Debian leadership team (O’Mahony and Ferraro, 2007) and with Fleming and Waguespack’s (2007) study of leadership on the Internet Engineering Task Force (IETF).

Maintaining one more package increased the likelihood of becoming a member of the NMC by 3.3%\(^8\). Meeting 5 more people (one standard deviation increase in degree centrality) increased the likelihood of becoming a member of the NMC by 76%. The popularity of one’s package is also marginally predictive of NMC status. For every 100 people who use a developer’s package, he or she is 4% more likely to become a NMT member. In 2002, these results are confirmed, even though the magnitude of the effect of degree centrality is smaller (Odds ratio=1.045)\(^9\). However, the number of packages and ties to the project leader is not significant in either model in 2002.

The negative effect of tenure may seem counterintuitive, but those who joined the project more recently were more aware of the problems of admitting new members.

\(^8\) To help the interpretation of the logit coefficients, the odds ratios are reported in parentheses. Odds ratio are computed by taking the antilogarithm of the logit coefficient, thus for the effect of the number of packages in 2001, we can simply compute: \(e^{\hat{b}_4}=1.04\). Values exceeding 1 indicate an increased likelihood of becoming a member of the NMC, while values less than 1 indicate decreased odds.

\(^9\) We also estimated other logit models using betweenness centrality, rather than degree centrality, and the results are consistent with the ones we obtained for degree centrality. These models are not reported in the paper but are available from the authors upon request.
Interviews with informants suggested that members who joined the project earlier were less likely to be interested in administration and more likely to be ‘hard core’ programmers interested in Linux prior to its commercialization.

*Narrowing the Pipeline.* Some members felt that the membership requirements established by the NMC were too onerous and undermined the freedoms Debian espoused. One informant expressed his gratitude at having joined the project before the new maintainer process was in place.

Raising the threshold too high, or even just the perceived notion, whether justified or not, that many NMs [New Maintainers] are unskilled, could make Debian more and more like an elitist society……As for me, I am just glad that I became a Debian developer over 3 years ago, long before this was even an issue. *(Posting to Debian Project, January 7, 2001)*

However, the process would become stricter yet. When processing candidates for membership, Application Managers found that many applicants were no longer interested or responsive. The head of the NMC proposed narrowing the pipeline of candidates by requiring applicants to obtain a sponsor before submitting their application.

An increasing number of applicants are either not serious about joining Debian and contributing to the project or not well prepared for the new maintainer process yet. A proposal is made to require all potential developers to maintain the recommendation of an existing developer…Those not very serious in joining are thus not able to apply in the first place *(Changes to the New Maintainer System, February 5, 2001)*.

This change, accepted as a means for members of the NMC to save time, effectively eliminated the possibility of newcomers without prior attachment to a network incumbent.

One project member (in the top 5% of maintainers in terms of the number of packages managed), running to become the fifth leader argued that the difficulties that underlay the new maintainer process were the product of Debian’s unusual success.
The biggest frustrations I see with Debian are, in fact, all related to this success. We have more developers than ever, more packages soaking more bandwidth to mirror than ever, more open bugs than ever, and our user community is broadening into areas where the criteria for success may be different. This puts enormous pressure on our organization, forcing us to continue to evolve…. (DPL Candidate Platform, February 23, 2001)

The winner of the 2001 election also proposed “adding more structure to the project”, namely to improve the new maintainer process. He noted that the NMC was under incredible pressure to protect Debian from ‘Trojans’ and that the current process was still not robust enough to handle further growth.

[We all have the same permissions to upload packages, we all have just as much right to screw up the archive as anyone else. In there lies the problem. Maintainer count is on the rise all the time. It may not seem to be a problem now, but increasing administrative and security work to keep this increase on a good footing is not going to remain easy (well, it isn’t easy now). Stopping developer entrance all together is not an alternative either. Some may argue that this makes Debian less "open". Well, I don’t think Debian is closed at all…..Now I know this isn’t the same as having one’s own packages, and that sponsorship is not turning out to be the godsend that we had hoped. However, allowing more levels of maintainship will likely make it easier for people to contribute. (Leadership Platform, February 20, 2001)

The DPL’s approach to keeping Debian open, but managing its boundaries, was to create differentiated levels of access to the code base, a change that was not implemented by the NMC.

Analyzing Network Expansion. With the implementation of the new maintainer process in 2001, the keyring network doubled in size to 532 nodes with 1212 ties. As entrance into the keyring required a face to face meeting, events such as the first project meeting held in France helped grow the keyring network. Several firms hired programmers to work on Debian and sponsored their travel to tradeshows and conferences. Informants reported that greater support for travel was a positive side effect of the growing commercial
support for Linux. Since developers do not live equidistantly and do not all receive corporate sponsorship, they likely face different probabilities of meeting each other. Thus, the Debian keyring network may not grow randomly. By analyzing the distribution of degrees (the number of people developers have met), we can determine if the Debian network is a random network.

If the network grows randomly, we would expect the degree distribution to approximate a Poisson distribution. If the network grows through a process of preferential attachment, where each new node is more likely to attach to a node with a higher degree, then it may follow a power law degree distribution where a large number of nodes has very few ties and a small number of nodes has a large number of ties (Albert and Barabasi, 2002; Clauset, Shalizi and Newman, 2009). Figure 2 reports the degree distribution of the Debian network. As an illustration, 77% of developers had met only 1 or 2 people in 1998 and 6% had met over 10. In 2002, 50% of developers had met 1 or 2 people and 4.8% had met over 25. Table 1 shows that the average degree of developers, stable from 1997-1999, increased to 3.64 in 2000 with a standard deviation of 4.67. This was much higher than the standard deviation in the two previous years. After 2000, the average degree of the network and its standard deviation increase every year10. Figure 3 presents the log-log plot cumulative degree distributions of the Debian keyring from 1997 until 2002 and confirms that the network does not follow a Gaussian or a Poisson distribution, but could be better described as a power-law distribution with an exponential cut-off11. Our research helps explain why this network follows this distribution: the NMC´s decision to require new members to be sponsored by an incumbent encouraged new members to attach to members with higher

---

10 If the network followed a power law distribution, a plot of the log number of degrees on the x axis and the log of the number of developers on the y axis would result in a straight line, and we can see from figure 3 that overtime the log-log plots approximates this distribution. This plot does not rule out a number of other types of distributions (log-normal, Weibull) but it is enough to argue that the tails of the distributions are “heavy” (Clauset, Shalizi and Newman, 2009).

---
degree centrality. This means that the gatekeeping position acquired by some developers, and institutionalized by the New Maintainer Committee is reinforced with the entry of new participants. At the same time the elite created by this process is not closed, and new contributors can become more central in the network and in the project.

Debian as a Bounded Entity. In 2001, Debian was regarded by the press as technically sophisticated, but lacking the reliability of a company. When reviewing Release 2.2, the editors of PC Magazine reported that “From a corporate standpoint, Debian’s main drawback is the lack of a company to support it…Debian’s developer community is very active, open and approachable, and Usenet groups and mailing lists are abundant, but don’t expect a lot of novice-level explanations. Commercial technical support for Debian is only available through third parties” (Ulrich, 2001). Debian remains non-commercial, but third parties can provide support and value added services12. Qualified developers can work on packages of their choosing, conditioned only by what their colleagues are doing. Debian’s development environment and source code remain publicly accessible. However, becoming a developer is no longer as easy as it once was. A Debian leader in 2003, agreed that it was much harder to become a member now than when he first joined the project, recounting the many tests a new member candidate must now pass to become member.

At the time new maintainer was not as formal or anything as it is now…[W]e are asking many more questions, and we’re doing more checks and everything. It is much more complicated now …for example you have to also look at a [traditional] license and say why that is not free software. We look at the different points in the Debian free software guidelines that it [a traditional license] violates…. [Y]ou have to agree that you comply with it [the GPL], and that you understand it…You have to summarize it and then you have to state explicitly that you agree to the social contract, and you have to explicitly agree that all that you do as part of Debian is free software as defined by the free software guidelines  (2003 DPL, June 20, 2003).

12 One such firm is www.projeny.com.
It can now take six months or more for a new maintainer to become a developer\textsuperscript{13}. Candidates are recommended to ask Debian developers to write a letter of reference for them – just as they would for a traditional bureaucratic organization. Applications are discussed in private and three types of rejection are possible: weak, strong and ultimate. An application manager’s decision to issue a weak rejection can be overturned by the NMC with \(\frac{1}{4}\) vote; a strong rejection can be overturned by a \(\frac{2}{3}\) vote and an ultimate rejection cannot be over turned\textsuperscript{14}. As of January 2009, 934 members have gone through the new process, 159 applications were in progress, and 24 applications were on hold\textsuperscript{15}. Forty-six application managers oversee this process. Fifty-nine people in 23 countries\textsuperscript{16} were looking to have their key signed despite the fact that 282 people in 39 countries offer to sign keys\textsuperscript{17}.

The new membership process reinforces conditions for network expansion through preferential attachment. Thus, new nodes will not enter randomly, but most likely attach to a node that already has a large number of ties. This can result in a sustained power-law distribution where a small number of nodes become even more highly connected and the bulk of nodes are only loosely connected. Since developers are more likely to attain gatekeeper positions if they are more connected, and new members tend to create new ties with developers with higher degree centrality, this pattern of growth and preferential attachment reinforces stability in gatekeeper positions. As suggestive evidence of this

\textsuperscript{13} In 2003, the NMC designed improvements to the New Maintainer Process to move registration, philosophy, procedures, task and skills test online. These improvements are expected to considerably speed the process.

\textsuperscript{14} Unfortunately, application rejection data was not available.

\textsuperscript{15} See \url{http://nm.debian.org} for more information.

\textsuperscript{16} These countries included Armenia, Argentina, Belgium, Brazil, Canada, Chile, China, Columbia, Germany, Denmark, Spain, Italy, Norway, Philippines, Poland, Portugal, Russian Federation, Singapore, Turkey, Ukraine United States, Vatican City State, and South Africa.

\textsuperscript{17} See \url{http://nm.debian.org/gpg_offer.php} and \url{http://nm.debian.org/gpg_need.php} for more information.
pattern, the leader of the New Maintainer Committee was elected Project Leader the following year.

**DISCUSSION**

“No topic more deeply engages scholars in the fundamental attributes of organizations” than the creation and revision of organizational boundaries (Santos and Eisenhardt, 2005: 505). Despite the fact that managing organizational boundaries is central to economics, organizational theory and sociology, observers of open communities producing complex knowledge goods have been overly fascinated with the very “openness” of these communities and neglectful of the ways in which these communities must somehow bound their activities in order to sustain their form. Organizational theorists have acknowledged that post-industrial organizations are no longer meaningfully bounded entities but comprised of porous and often shifting boundaries and that this has widened the gap between theory and post-industrial modes of organizing: “Explanation through counting misses the major dynamics of the new economy” (Davis and McAdam, 2000: 205). Thus, our methods, approaches and vocabularies need to be enriched in order to better understand and explain emerging organizing phenomena that may not depend upon a shared place or context to support production (Barley and Kunda, 2001).

One empirical strategy is to examine processes held in common across a variety of new organizing contexts (Lamont and Molnar, 2002). We pursued such an approach by focusing on a particular boundary dispute: how to ascertain who becomes a member to a new organization. This dispute is one that is central to collectively managed organizations: “How is collective action coordinated when participation by ‘members’ is impromptu and impermanent?” (Davis and McAdam, 2000: 214). Our research examined this boundary decision in the context of a collectively managed open source software project that had no
prior structures in place to help determine how new members should be brought into the organization. By observing how the Debian community confronted this challenge, we learned: 1) how one type of organizational boundary is established, 2) the behaviors that are likely to predict who becomes a gatekeeper of this system; and 3) identified a critical mechanism that affects future growth of the project’s social network. Together, our research helps deepen our understanding of how social networks and new organizing structures co-evolve.

Explosive growth in contributors to the project led to an organizational crisis upon which the project closed its doors to all new members - until a group of project incumbents could design a new process that would ensure that prospective members’ skills, goals, and ideology were aligned with the collective. Even though some within the project displayed reluctance to “close” the project in any way, our field research shows how the community came to terms with this crisis – forming a New Maintainer Committee to establish rules and a process for membership and thus becoming future gatekeepers to the project. This new process makes organizational boundaries more difficult to penetrate over time – narrowing but not closing. With statistical analyses, we discovered that those project contributors that had met more people face-to-face and were more central in the project’s social network were more likely to become gatekeepers to the project. While project members coordinate their work almost solely online, when choosing individuals to manage organizational boundaries, they trust those they have met.

Further analysis of the network’s expansion shows that the project’s network does not grow randomly but develops through preferential attachment, leading to a power law distribution. Our field data helps explain how preferential attachment operated in this community. The New Maintainer Committee designed a process that required sponsorship
from a project incumbent and a face-to-face meeting. These rules set in motion a process whereby new entrants are more likely to connect with project members that are more central than with any random project member. Thus, the committee designed a membership process that reinforced their role as gatekeepers who would guide future expansion of the network. The boundary dispute, in essence, was resolved with a membership process that reshaped the subsequent social structure of the community.

Together, these three contributions help explain the mechanisms that permit scale-free social networks to come into existence. Our research approach shows that, as suggested by Lamont and Molnar, 2002) there is value in studying boundary disputes. However, we took this charge a step further by using boundary disputes as way to unpack our understanding of how networks and organizational structures co-evolve over time. Network processes shape who emerges to define organizational boundaries; how those boundary rules are set then affects the project’s future network structure. Our research shows how networks evolve to affect the design of organizational practices and policies which can then, in turn, affect the future structure of the network.

Prior research on the evolution of networks in the biotechnology industry found that no single rule of attachment dominated the creation of new ties. The recombinative nature of innovation in the biotechnology field demands a constant quest for new partners and ideas, with actors in this field preferring diverse ties as opposed to connecting with incumbents or similar others (homophily) (Powell et al, 2005). However, this same study tracked the emergence of an ‘open elite’ where those that were more central continuously refreshed themselves by welcoming new entrants. Similarly, our research of Debian’s evolving organization shows that elites define the terms of admittance. In doing so, they help the project to stay open, but bounded, sustaining their own structural advantage. In terms of
the on-going boundary dispute between open and commercial science, our research suggests that those in the position to make boundary rules can affect the networks and social structure that follow. For example, if patent holders acquired more gatekeeper roles in academia, our research suggests that this would likely shape the management of boundaries between open and commercial science.

This research has broader implications for researching collectively managed knowledge communities. First it suggests that literature on virtual communities and new organizational forms remains naïve to the realities and challenges of producing complex knowledge goods in a distributed community. New organizational forms and virtual organizations are often depicted as nebulous and constituted by a shifting and ever changing body of members. However, volatility and turnover in new organizational forms is not the same as indeterminacy; flow is not the same as ambiguity and these distinctions have different implications for the study of organizational boundaries.

Boundary definition and management is likely to be critical for any type of community that strives to remain open while producing knowledge goods, but must do so without jeopardizing the security and stability of their work – as David and Foray so presciently predicted. “What is at stake here is the entire range of mechanisms that will facilitate interpersonal and inter-organizational transactions, given the new conditions for knowledge transactions and exchanges…Clearly new methods need to be devised to "certify" the knowledge circulating on the Internet” (David and Foray, 2002: 17). If knowledge producing communities that cross or exist outside of established organizational boundaries are to survive, innovations in trust and reliability mechanisms will continue to emerge and require further comparative study.
More research on mechanisms that help carve a protected and common informational space for collaboration (O’Mahony, 2003) is needed to understand not only the emergence of such mechanisms, but their contribution to project security and effectiveness. We should point out that security and effectiveness can be measured in multiple ways. In the case of Debian, new membership criteria tested not only technical quality, but new entrants’ knowledge of software licenses in order to ensure that contributions received from new members would comply with the project’s legal boundary with the commercial world (see also Chen & O’Mahony, 2009). In this sense, the membership boundary is one way in which the larger open source and industry boundary dispute is managed.

Research on mechanisms should attend to the crises such communities confront; the triggers of such crises; as well as the range of solutions from which communities draw to address these crises. Research on how communities scale is not popular but, as David and Foray point out, the challenge of knowledge production “become[s] greater as a community expands” as does “costs of data search, the risk of congestion and anonymity amongst members, which can in turn, represent a source of acute problems of trust” (David and Foray, 2002: 8). Our research shows that problems of trust and malfeasance are very real and that the design of new mechanisms to manage them will co-evolve with the social networks that underlie knowledge producing communities.
References


O’Mahony, Siobhan. (2003). Guarding the Commons: how community managed projects
protect their work, Research Policy (32): 1179-1198.


The shaded area indicates the time during which the Debian project closed its doors to new members.

*Source:* Data on articles citing “Debian Linux” was collected from ABI/Inform database in Proquest and the Factiva database in 2003.
Fig. 2 Degree distribution of the Debian Network, 1997-2002.

1997

1998

1999

2000

2001

2002

Fig. 3 Log-log plot of the cumulative degree distribution of the Debian Network, 1997-2002
Table 1: Growth in the DebianKeyring Network

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Developers in network</td>
<td>13</td>
<td>82</td>
<td>176</td>
<td>298</td>
<td>532</td>
<td>671</td>
</tr>
<tr>
<td>Growth rate</td>
<td></td>
<td>530.77%</td>
<td>114.63%</td>
<td>69.32%</td>
<td>79.32%</td>
<td>26.13%</td>
</tr>
<tr>
<td>Number of ties</td>
<td>11</td>
<td>111</td>
<td>239</td>
<td>543</td>
<td>1212</td>
<td>2014</td>
</tr>
<tr>
<td>Average Degree</td>
<td>2.46</td>
<td>2.71</td>
<td>2.72</td>
<td>3.64</td>
<td>4.56</td>
<td>6.00</td>
</tr>
<tr>
<td>S.D. Degree</td>
<td>1.60</td>
<td>3.04</td>
<td>3.22</td>
<td>4.67</td>
<td>6.57</td>
<td>8.84</td>
</tr>
<tr>
<td>Number of Components (min 2)</td>
<td>3</td>
<td>19</td>
<td>29</td>
<td>31</td>
<td>39</td>
<td>33</td>
</tr>
<tr>
<td>Density</td>
<td>0.124</td>
<td>0.033</td>
<td>0.015</td>
<td>0.012</td>
<td>0.009</td>
<td>0.009</td>
</tr>
</tbody>
</table>

Table 2: Descriptive statistics of Debian Developers, 2001-2002.

<table>
<thead>
<tr>
<th></th>
<th>2001</th>
<th>2002</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>All Developers (N=532)</td>
<td>All Developers (N=671)</td>
</tr>
<tr>
<td></td>
<td>Mean</td>
<td>S.D.</td>
</tr>
<tr>
<td>Dependent Variable</td>
<td></td>
<td></td>
</tr>
<tr>
<td>New MaintainerCommitteea</td>
<td>.010</td>
<td>.005</td>
</tr>
<tr>
<td>Independent Variables</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of Packages maintained</td>
<td>6.68</td>
<td>9.01</td>
</tr>
<tr>
<td>Package popularity</td>
<td>299.5</td>
<td>809.8</td>
</tr>
<tr>
<td>Tenure (in months)</td>
<td>18.54</td>
<td>15.50</td>
</tr>
<tr>
<td>Europe (reference category)a</td>
<td>.46</td>
<td>.47</td>
</tr>
<tr>
<td>North-Americaa</td>
<td>.31</td>
<td>.31</td>
</tr>
<tr>
<td>Other continenta</td>
<td>.12</td>
<td>.13</td>
</tr>
<tr>
<td>Tie to Leadera</td>
<td>.11</td>
<td>.12</td>
</tr>
<tr>
<td># of mailing list postings</td>
<td>12.24</td>
<td>28.43</td>
</tr>
<tr>
<td>Degree Centrality</td>
<td>4.56</td>
<td>6.58</td>
</tr>
</tbody>
</table>

*aDummy variables
Table 3: Correlation coefficients in 2001.

<table>
<thead>
<tr>
<th>Variables</th>
<th>(1)</th>
<th>(2)</th>
<th>(3)</th>
<th>(4)</th>
<th>(5)</th>
<th>(6)</th>
<th>(7)</th>
<th>(8)</th>
<th>(9)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) New Maintainer committee</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(2) # of packages maintained</td>
<td>0.1568</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(3) Package popularity</td>
<td>0.1134</td>
<td>0.3759</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(4) Degree centrality</td>
<td>0.3152</td>
<td>0.1359</td>
<td>0.1203</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(5) Tie to Leader</td>
<td>0.1383</td>
<td>0.0540</td>
<td>0.1017</td>
<td>0.4655</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(6) # of mailing list postings</td>
<td>0.1069</td>
<td>0.2852</td>
<td>0.3754</td>
<td>0.3199</td>
<td>0.1359</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(7) Tenure (months)</td>
<td>-0.0548</td>
<td>0.0075</td>
<td>0.1501</td>
<td>0.2781</td>
<td>0.2014</td>
<td>0.1021</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(8) North-America</td>
<td>-0.0161</td>
<td>-0.0652</td>
<td>0.0176</td>
<td>-0.1058</td>
<td>-0.0477</td>
<td>0.0489</td>
<td>-0.0918</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>(9) Others</td>
<td>-0.0071</td>
<td>0.0786</td>
<td>0.0405</td>
<td>-0.0724</td>
<td>-0.1022</td>
<td>0.0166</td>
<td>-0.0518</td>
<td>-0.2506</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 4: Correlation coefficients in 2002.

<table>
<thead>
<tr>
<th>Variables</th>
<th>(1)</th>
<th>(2)</th>
<th>(3)</th>
<th>(4)</th>
<th>(5)</th>
<th>(6)</th>
<th>(7)</th>
<th>(8)</th>
<th>(9)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) New Maintainer committee</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(2) # of packages maintained</td>
<td>0.0846</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(3) Package popularity</td>
<td>0.0751</td>
<td>0.3738</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(4) Degree centrality</td>
<td>0.0725</td>
<td>0.0796</td>
<td>0.0892</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(5) Tie to Leader</td>
<td>0.0076</td>
<td>0.0177</td>
<td>0.0982</td>
<td>0.3513</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(6) # of mailing list postings</td>
<td>0.0641</td>
<td>0.2208</td>
<td>0.3752</td>
<td>0.2787</td>
<td>0.0410</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(7) Tenure (months)</td>
<td>-0.1145</td>
<td>-0.0364</td>
<td>0.1552</td>
<td>0.1805</td>
<td>0.2396</td>
<td>0.0430</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(8) North-America</td>
<td>0.0568</td>
<td>-0.0391</td>
<td>0.0180</td>
<td>-0.1453</td>
<td>-0.0385</td>
<td>-0.0172</td>
<td>-0.0710</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>(9) Others</td>
<td>-0.0422</td>
<td>0.0804</td>
<td>0.0361</td>
<td>-0.1035</td>
<td>-0.0752</td>
<td>0.0608</td>
<td>-0.0647</td>
<td>-0.2630</td>
<td>-</td>
</tr>
</tbody>
</table>
Table 5: Logistic Regression coefficients for the regression of New Maintainer Committee membership on selected independent variables in 2001.

<table>
<thead>
<tr>
<th></th>
<th>(1)</th>
<th>(3)</th>
<th>(5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Packages</td>
<td>0.035**</td>
<td>0.033**</td>
<td>0.033**</td>
</tr>
<tr>
<td></td>
<td>(1.036)</td>
<td>(1.034)</td>
<td>(1.033)</td>
</tr>
<tr>
<td>Package Popularity</td>
<td>0.0002</td>
<td>0.0003</td>
<td>0.0004*</td>
</tr>
<tr>
<td></td>
<td>(1.0002)</td>
<td>(1.0003)</td>
<td>(1.0004)</td>
</tr>
<tr>
<td>Tenure in the Project</td>
<td>-0.027**</td>
<td>-0.028**</td>
<td>-0.049***</td>
</tr>
<tr>
<td></td>
<td>(0.974)</td>
<td>(0.973)</td>
<td>(0.952)</td>
</tr>
<tr>
<td>Tie to leader &lt;br&gt; (year t-1)</td>
<td>1.194***</td>
<td>1.166***</td>
<td>0.170</td>
</tr>
<tr>
<td></td>
<td>(3.299)</td>
<td>(3.211)</td>
<td>(1.186)</td>
</tr>
<tr>
<td>North America a</td>
<td>-0.095</td>
<td>-0.133</td>
<td>0.153</td>
</tr>
<tr>
<td></td>
<td>(0.910)</td>
<td>(0.876)</td>
<td>(1.165)</td>
</tr>
<tr>
<td>Other Continent a</td>
<td>-0.139</td>
<td>-0.152</td>
<td>0.138</td>
</tr>
<tr>
<td></td>
<td>(0.870)</td>
<td>(0.859)</td>
<td>(1.147)</td>
</tr>
<tr>
<td>Number of Postings &lt;br&gt; (year t-1)</td>
<td>0.003</td>
<td>-0.001</td>
<td>(0.999)</td>
</tr>
<tr>
<td></td>
<td>(1.003)</td>
<td>(0.999)</td>
<td></td>
</tr>
<tr>
<td>Degree Centrality</td>
<td>0.142***</td>
<td></td>
<td>(1.152)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intercept</td>
<td>-2.292***</td>
<td>-2.303***</td>
<td>-2.686***</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Log-likelihood ratio</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>for model estimated:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>vs. null model</td>
<td>23.73††</td>
<td>24.89††</td>
<td>57.63††</td>
</tr>
<tr>
<td></td>
<td>(6)</td>
<td>(7)</td>
<td>(8)</td>
</tr>
<tr>
<td>vs. previous model</td>
<td>1.16</td>
<td>32.74††</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(1)</td>
<td>(1)</td>
<td></td>
</tr>
<tr>
<td>Observations</td>
<td>515</td>
<td>515</td>
<td>515</td>
</tr>
<tr>
<td>Pseudo R-squared</td>
<td>0.07</td>
<td>0.07</td>
<td>0.17</td>
</tr>
</tbody>
</table>

Odds ratios in parentheses
a Compared to developers located in Europe
*=p<0.1, **=p<0.05, ***=p<0.01 (one tailed tests)
† χ² significant at the level (p = <.05)
†† χ² significant at the level (p = <.01)
Table 6: Logistic Regression coefficients for the regression of New Maintainer Committee membership on selected independent variables in 2002

<table>
<thead>
<tr>
<th></th>
<th>(1)</th>
<th>(2)</th>
<th>(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of packages</td>
<td>0.025</td>
<td>0.022</td>
<td>0.024</td>
</tr>
<tr>
<td></td>
<td>(1.025)</td>
<td>(1.023)</td>
<td>(1.024)</td>
</tr>
<tr>
<td>Package Popularity</td>
<td>0.0004*</td>
<td>0.0003*</td>
<td>0.0004*</td>
</tr>
<tr>
<td></td>
<td>(1.0004)</td>
<td>(1.0003)</td>
<td>(1.0004)</td>
</tr>
<tr>
<td>Tenure in the Project</td>
<td>-0.060***</td>
<td>-0.060***</td>
<td>-0.069***</td>
</tr>
<tr>
<td></td>
<td>(0.942)</td>
<td>(0.942)</td>
<td>(0.933)</td>
</tr>
<tr>
<td>Tie to leader (year t-1)</td>
<td>0.662</td>
<td>0.673</td>
<td>0.381</td>
</tr>
<tr>
<td></td>
<td>(1.939)</td>
<td>(1.959)</td>
<td>(1.463)</td>
</tr>
<tr>
<td>North America a</td>
<td>0.397</td>
<td>0.398</td>
<td>0.616</td>
</tr>
<tr>
<td></td>
<td>(1.487)</td>
<td>(1.489)</td>
<td>(1.852)</td>
</tr>
<tr>
<td>Other Continent a</td>
<td>-0.799</td>
<td>-0.817</td>
<td>-0.556</td>
</tr>
<tr>
<td></td>
<td>(0.450)</td>
<td>(0.442)</td>
<td>(0.573)</td>
</tr>
<tr>
<td>Number of Postings (year t-1)</td>
<td>0.002</td>
<td>0.001</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(1.002)</td>
<td>(1.001)</td>
<td></td>
</tr>
<tr>
<td>Degree Centrality</td>
<td></td>
<td></td>
<td>0.044**</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(1.045)</td>
</tr>
<tr>
<td>Intercept</td>
<td>-2.442***</td>
<td>-2.481***</td>
<td>-2.717***</td>
</tr>
</tbody>
</table>

Log-likelihood ratio for model estimated:

<table>
<thead>
<tr>
<th></th>
<th>(1)</th>
<th>(2)</th>
<th>(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>vs. null model</td>
<td>21.52††</td>
<td>22.31††</td>
<td>27.11††</td>
</tr>
<tr>
<td>(df)</td>
<td>(6)</td>
<td>(7)</td>
<td>(8)</td>
</tr>
<tr>
<td>vs. previous model</td>
<td>2.37</td>
<td>4.8†</td>
<td></td>
</tr>
<tr>
<td>(df)</td>
<td>(1)</td>
<td>(1)</td>
<td></td>
</tr>
<tr>
<td>Observations</td>
<td>647</td>
<td>647</td>
<td>647</td>
</tr>
<tr>
<td>Pseudo R-squared</td>
<td>0.09</td>
<td>0.09</td>
<td>0.11</td>
</tr>
</tbody>
</table>

Odds ratios in parentheses

* = p < 0.1, ** = p < 0.05, *** = p < 0.01 (one tailed tests)
† † = $\chi^2$ significant at the level (p = .05)
** = $\chi^2$ significant at the level (p = .01)

Compared to developers located in Europe