

Managing the Boundary of an 'Open' Project

Siobhán O'Mahony
Harvard University
somahony@hbs.edu

Fabrizio Ferraro
IESE Business School, University of Navarra
fferraro@iese.edu

In *Market Emergence and Transformation*, edited by John Padgett and Walter Powell,
forthcoming.

February 25, 2005

This research was supported by Stanford University Center for Work, Technology, and Organization, the Social Science Research Council Program on the Corporation and the Harvard University Business School Division of Research. We thank the attendees of the Santa Fe Workshop on the Network Construction of Markets for their helpful comments, in particular the helpful guidance of Woody Powell, Steve Barley and Ben Adida. The data collection and processing efforts of John Sheridan and Vikram Vijayaraghavan 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 open source software projects with porous boundaries develop code in an open and public environment. This research uses a multi-method approach to understand how one community managed open source software project, Debian, develops a membership process. We examine the project's face-to-face social network during a five-year period (1997-2002) to see how changes in the social structure affect the evolution of membership mechanisms and the determination of gatekeepers. While the amount and importance of a contributor's work increases the probability that a contributor will become a gatekeeper, those more central in the social network are more likely to become gatekeepers and thus influence the membership process. A greater understanding of the mechanisms open projects use to manage their boundaries has critical implications for knowledge producing communities operating in pluralistic, open and distributed environments. It also contributes to our theoretical understanding of how network structures help shape the construction of new social orders.

Observers of the open source phenomenon often compare the production and management of open source code 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 with informed skepticism (Merton, 1973; Latour & Woolgar, 1979). While this comparison may be only partially true, both open science and open source software face similar challenges with respect to preserving the boundary between the open and public production of knowledge and the commercial capturing of value. Since the passage of the Bayh Dole Act, scholars have increasingly examined how expanded capabilities for licensing academic research have affected scholarship and the intellectual commons. Since the term open source was created in 1998, free and open source software has attracted a growing commercial market and a more diverse group of supporters: testing their ability to remain a loose organic form.

Open source software is conducted in public and, like scientists, contributors treat code from others with informed skepticism. However, open source projects do not have the socialization processes and institutional rigors shared by open science. Academia has a long history indoctrinating graduate students to the norms of open science that open source projects do not share. How do open source projects ensure that incoming contributors do not violate their mission of creating open code?

We propose that open projects are not without their boundaries and that by examining how open source software projects manage their boundaries, we can learn more about how knowledge producing communities conceptualize, enact and sustain their boundaries with the commercial world. A growing body of work on social boundaries (Lamont and Molnar, 2002) suggests that the construction, revision, enactment, and destruction of boundaries across areas as diverse as class, gender, communities, nations, territory, professions, science and knowledge may have common processes. This research

contributes to our understanding of boundary processes in the creation of science and knowledge but has applications to these other areas as well.

There are several ways in which open source projects can protect their boundaries. In this paper, we focus on one: designing a membership process. With qualitative data, we unpack the types of threats faced by open projects and the types of mechanisms they design to manage these threats. With quantitative data, we analyze the structure of the network to assess what determines who becomes a designer of the new membership process, and thereby gatekeepers to the project. We then examine how this affects the structure of the network. By doing so, we contribute to a theoretical understanding of how networks and social structures co-evolve.

Managing the Boundary Between Open and Commercial Science

Many legal, economic and organizational scholars have become concerned that critical distinctions between public (or ‘open’) and commercial science have become blurred (Dasgupta and David, 1994). The realms of open and commercial science are interdependent but distinct knowledge systems that 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 is characterized by norms that encourage universalistic standards based on competence or merit (Merton, 1973). The practice of open science demands full disclosure of methods and findings, to allow scientists to replicate and verify each other’s work (Merton 1973; Latour & Woolgar, 1979). While the goal of commercial investment in 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 remains fragile and can be undermined if too great an emphasis is placed on property right protections (2000). Thus, more scholars are examining how the Bayh Dole Act, which permits an expanded role for universities in licensing their 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 public science are compromised when university endeavors cross scholarly and commercial boundaries.

There is suggestive evidence that the blurring of commercial and university science boundaries has, in the last twenty years, had some effects. Between 1981 and 1998, Owen-Smith found that science conducted at 89 of the most research intensive US universities became increasingly affected by commercial concerns, standards and rewards (2003). Careful empirical analysis shows that university patenting experience positively affected the visibility or impact of academic work after 1983 (Owen-Smith, 2003). Further examination of these same 89 US universities shows that university technology licensing offices play a critical role in assessing the commercial impact of basic research (Owen-Smith and Powell, 2003) in effect, performing a boundary spanning role (Owen-Smith, 2005). Universities that were 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 in orientation due to the management of boundaries with commercial entities simultaneously highlights the importance and fragility of the balance between public and private institutional frameworks in the production of new knowledge. 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 regime can prosper without unduly influencing academic research in a direction that is not particular to one firm. Sustaining openness and pluralism without risking cooptation from commercial entities is a central concern. This tension is one with which open source projects are also gaining more experience.

Managing the Boundary between Open Source and Commercial Software

Since commercial firms began contributing to open source and free software, investors and the media have sustained a steady interest in its production. This phenomenon, coupled with expansion in global and public sector markets, has attracted a new population interested in contributing to or selling open source software. Some community managed open source software projects have been receptive to this expanded commercial audience and have engaged in synergistic relations with firms and governments, but yet remain wary that their culture, practice, and code may be compromised (O'Mahony, 2002). Sponsorship from industry dominants has introduced a new challenge: how to maintain vendor neutrality in the face of increasing industry support (O'Mahony, 2002).

But, open source projects lack the strong institutional and professional structures academe provides to guide training and access to the profession. Legal protections help. Software protected by licenses such as the GNU General Public License (GPL), assures potential collaborators that their efforts will not, in the future, become absorbed into

proprietary software (Stallman, 1999). Other works derived from GNU GPL licensed software must be guided by the same terms. This foundation provides a common platform for collaboration¹. However, open source projects that are community founded are not managed with formal authority relations. Although open source communities are guided by powerful norms that reinforce or discourage certain types of online behavior, few have deep experience developing formal social structures.

Communal and directly democratic organizations often have difficulty scaling (Whyte & Whyte, 1988; Rothschild-Whitt, 1979; Rothschild and Russell, 1986; Rothschild and Whitt, 1986), but community managed projects face a more complex dilemma. For community managed projects to sustain themselves, not only must their informal social structures mature to support more contributors, but they must also ensure pluralism while encouraging collective action toward a common goal. Unlike cooperatives, unions, kibbutz, and social movement organizations that often reinforce homogeneity among their members (e.g. Kanter, 1968), community managed projects derive their technical and social strength from the diversity of the social, geographic and technical environments of their contributors. Differences in talents, opinions, and computing environments helps contributors identify unique problems and solutions that fosters a more robust technical product - that is one that can be used reliably by a broader user base (Tuomi, 2003; Waynor, 2000; Raymond, 1999).

Diversity in perspectives enhances the resilience of community developed software, but contributors must share a common goal in the project's success as a non-commercial entity. As the number of contributors to community managed projects increases, so too does the diversity of contributor skill and motivation. In a completely open environment,

¹ In reaction to the increased patentability of their work, some scientists and engineers have also developed sophisticated tactics to protect the areas they are working on from becoming appropriated. Such tactics include placing work in the public domain, defensive patenting, and offering rewards to research and identify prior art to invalidate patents.

not all contributors may have the project's best interests in mind. And, good intentions without skill can be equally dangerous. Under these conditions, maintaining the quality of code and ascertaining the motivations of contributors has become a concern for some community managed projects (Michlmayr and Hitt, 2003). The potential for someone to co-opt or hijack a project or unwittingly introduce code owned by someone else looms large.

While open source projects are 'open' in both process and product, recent research has attended to the ways in which such projects are also bounded. For example, despite the popular belief that open source contributors give their work away, many contributors to large, successful open source projects actually own their own work and 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.

Large successful open source projects struggle to remain open to new contributors and develop code in a pluralistic and public environment that still assures them of the source and quality of contributions received. Quality in this context is not merely technical but also legal, referring to code that is free of proprietary ownership rights. Project members want to ensure that potential contributors share the project's technical and legal values and do not introduce code that might jeopardize the project's boundary with proprietary software or conflict with their licenses. Thus, managing project membership affects not only the boundary of the project community itself, but is a critical component to preserving the project's boundary with the commercial world.

On an open source project, the vast majority of coding activities are publicly accessible and the software produced can be downloaded for free. But, access rights to the code base must be managed so as not to jeopardize its security. Unlike virtual organizations or online communities with fluid boundaries and potentially anonymous, shifting members and identities, members of open source projects must maintain a trusted identity in order to gain the ability to contribute directly to a project's code base. Membership can be fluid, but it cannot be *indeterminate*. Members may not always maintain consistent activity rates, but the allocation of access rights must be known and distinguishable. Since contributors may never meet each other, they face a unique problem: how to verify the identities of individuals distributed around the world. Powell (1990) theorized that network forms may face novel problems of control and that membership in a community may require new organizational practices (Powell et al, 1996: 142). Our research on community managed projects identifies a novel array of practices designed to foster trust and identity authentication.

Identity Authentication: Cryptography and the “Web of Trust”

Some technical communities have developed a means to secure member identities using public key cryptography, thus providing a unique source of network data. Cryptography uses mathematical algorithms to encode data so that it can travel across insecure networks and be 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 cryptography solves 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 prior key distribution. Thus, a user's private key is

never revealed (Network Associates, 1990). 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.

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 it does not verify the link between the key and the sender’s identity. You can prove mathematically that only the owner of the private key can perform a particular operation, but how do you prove ownership of the private key? To make public-key cryptography useful, a real-world identity must be linked to a given public key. Several technical communities use the practice of ‘key signing’ in order 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) is required to show that a public key belongs to owner and is represented by the user id packet (Brennen, 2003)².

In a globally distributed environment, where everyone cannot meet everyone else, responsibility for validating public keys is delegated to trusted others. In programming communities, key signers are explicitly encouraged to consider not only their own security requirements but, also 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).”

² 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).

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.

One way individuals have their keys signed is by hosting or attending a 'key signing party': a get-together 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).

[P]lease 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 members of the group. Signing someone's key without physical verification of his or her identity breaches the norms of the community and threatens the validity of 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.

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 its social structure. Using cryptography data from the keyring of the Debian project, we examine the evolution of its social network over a five-year period (1997 – 2001) to assess how changes in its structure affect 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 of the Linux operating system 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 programmers³ distributed in over 40 countries who collectively maintain over 8,000 software packages.

The project's tenure, technical and organizational success provides a unique forum to study the evolution of their social network and development of mechanisms to manage the project's boundaries over time. With qualitative data on the project's evolution we examine the membership crisis Debian experienced and their approach to managing a potential influx of contributors new to the project's norms, methods, and values. A gatekeeping role responsible for designing a formalized membership process emerges. With analysis of the project's network data, we find that one's structural position affects the attainment of gatekeeper positions. Furthermore, project members who obtain these positions, by influencing the membership process, may reinforce preferential attachment mechanisms that affect future growth of the network. These results confirm that organizational design and

³ 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.

social network dynamics are intertwined and that the former cannot be ignored when attempting to understand the emergence of new social orders.

Methods

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. Since each key signing is dated and requires a face to face meeting, these data indicate when individual project members met each other. 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 became 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.

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 by computing the *package popularity*. Since early 2003, Debian users could install a "popularity-contest package" that automatically calculates the number of people using 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, debian devel.

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 met face-to-face at least once⁴. Finally, we created a dummy variable indicating whether a developer had met and 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.

In order to understand this 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. We

⁴ 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.

also used Pajek software (Batageli and Mrvar, 1998)⁵ to generate a replicable visualization of the network of Debian developers. The drawings were constructed with the Kamada-Kawai algorithm, which locates connected nodes adjacent to one another and sets the physical distance between nodes as a function of the shortest path between them. The distances between unconnected portions of the network are undefined in this algorithm, and therefore the relative position of different components does not have substantive meaning (Kamada-Kawai, 1989). To show how the structural position of the developers affected the composition of the New Maintainer Committee, we estimated a logit model (Long, 1997), testing whether degree centrality in the network affected this outcome in 2001-2002, after controlling for the level and criticality of contribution, ties to the project leader, tenure in the project and geographic location.

Debian's Evolving Membership Process

Project Initiation. On August 16, 1993, the founder of Debian proposed developing an easily installable packaged version of the GNU⁶/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,⁷ Debian is being developed openly in the 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

⁵ This software, developed by Vladimir Batagelj and Andrej Mrvar, is freely available for noncommercial use, and can be downloaded at <http://vlado.f.f.uni-lj.si/pub/networks/pajek/>. See de Nooy, Mrvar, and Batageli (2003) for an introductory text to exploratory social network analysis with Pajek.

⁶GNU 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.

⁷ This reference to other Linux distributions managed by one person likely refers to the Linux kernel managed by Linus Torvalds.

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.”⁸ 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 multilateral contributions. 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; 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

⁸ The origins of the project’s name stems from a combination of the founder and the founder’s wife’s names.

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 had resigned at the end of 1996 angrily defended improving 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. 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 purposely 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, thus 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.⁹ 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.

⁹ 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. (*Interview, 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 of 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 members' 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 detailed specific privileges for members without articulating how one would become 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 access to the code repository and in order to upload a package directly, 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. (*Interview, 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 (*Interview, Sponsored Contributor, March 20, 2001*).

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

To address the problem of accepting unqualified members, the fourth project leader 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 Debian maintainership 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 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 to have confidence in their philosophical commitment above and beyond their degree of 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, the person’s current [employment] situation, as well as [the person’s] long term contact information must be clear” (*New Maintainer Proposal, October 17, 1999*). Project members engaged in this debate recognized that face-to-face meetings could help instill greater collegiality, trust and respect among members, even enforcing some normative control.

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 (I suspect I'll be better behaved on the lists now I've met a few of you at ALS, for example ;) *(Posting to Debian Project Mail list, October 18, 1999).*

This developer felt that he was better behaved having met others on the project.

The 'internship period' 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 would conflict with the project and affect its non-commercial status. This proposal led to a discussion of what it meant to be an 'open project' with a Constitution that places no restrictions on its members. 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 architects of the network. Who is more likely to become a gatekeeper? To answer this question, we examine how the structure of the network affected the probability of joining the NMC. Table 5 presents the logit coefficients for models predicting membership on the 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 2002¹⁰. We also control 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 has a positive significant effect on the likelihood of joining the NMC in 2001, while tenure has a negative effect. In the second model for 2001, we include the number of mail list postings made the year prior and find no effect. In the third model for 2001, we include degree centrality and find a positive significant effect that is larger than any other variable.

¹⁰ The correlation coefficients of the variables are reported, for 2001 and 2002, in tables 3 and 4.

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. Maintaining one more package increased the likelihood of becoming a member of the NMC by 3.3%¹¹. 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 mostly confirmed, even though the magnitude of the effect of degree centrality is smaller (Odds ratio=1.045)¹². However, the number of packages and ties to the project leader is not significant in either model in 2002.

Our findings show that technical contributions are predictive up to a point, but degree centrality has a significant and positive influence on becoming a gatekeeper. In Fleming and Waguespack's study (2003) study of leadership on the Internet Engineering Task Force (IETF), technical contributions were also predictive of leadership, but including the 'reflected status' of others (as measured by publications of their colleagues) provided a stronger predictor of leadership. While the significance of one's code contributions does have an effect in our study, having more ties in the network seems to matter more to become a member of the membership committee.

Tenure likely had a negative effect as newcomers to the project may have been more aware of the problems of admitting new members. Project members who joined earlier in

¹¹ 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: $e^{0.04}=1.04$. Values exceeding 1 indicate an increased likelihood of becoming a member of the NMC, while value less than 1 indicate decreased odds.

¹² 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.

the project may also have been less interested in administration and devoted solely to coding. Interviews with informants suggest that project old timers were more likely to have been 'hard core' programmers interested in Linux prior to its commercialization in the market.

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 to by 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 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.

[W]e 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 with a narrower pipeline, was to create different levels of access to the code base, a change that was not implemented by the NMC.

Analyzing Network Expansion. With implementation of the new maintainer process in 2001, the keyring network doubled in size to 532 nodes with 1212 ties. 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 then we would expect the degree distribution to approximate a Poisson distribution (Albert and Barabasi, 2002). 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 (Albert and Barabasi, 2002). Networks with a power law distribution are scale-free in that a large number of nodes have very few degrees and a small number of nodes have a large number of degrees.

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 year. If the network is scale-free, a plot of the log number of degrees on the x axis and the log of the number of developers on the y axis will result in a straight line. Figure 3 presents the log-log plot degree distribution of the Debian keyring from 1997 to 2002 and illustrates that the power-law is in effect.

While Barabasi and Albert argue that power law distributions fueled by preferential attachment can be found in natural, technical, and social systems (1999), scholars of complex networks have not found universal mechanisms to explain preferential attachment (2002). Thus, how such processes are enacted at the micro-level in social systems is unclear. Our qualitative study helps explain why. The NMC's decision to require new members to be sponsored by an incumbent may have encouraged new members to attach to members with higher degree centrality.

Another way to analyze the growth of the network is through network graphic visualization techniques. Figures 4, 5, and 6 illustrate the growth of the Debian network from 2000 – 2002. Vertices were scaled to represent the number of packages a developer maintains. If centrality in the network was affected by the number of packages maintained we would expect to see larger nodes at the center of the network. Instead, several of the largest nodes are at the edges of the network. This indicates that there are major contributors who have not met many other people and are not central to the network. Over the next two years (Figures 5 and 6), there does not appear to be a substantial relationship between node size and position in the structure of the network. The presence of many large nodes at the periphery suggests that some of the heavy lifters on the project are less interested in the practice of meeting others and having their key signed.

In these same figures, the nodes are color coded to indicate when individuals entered the network. The color yellow indicates developers who entered the network in 1996 or earlier, green in 1997, red in 1998, blue in 1999, grey in 2000, and silver in 2001. If there was a strong relationship between one's tenure and one's centrality in the network, than we would expect to see those who arrived earliest in the network in the center. The 2000 network (Figure 4) would have yellow nodes at the center surrounded by concentric rings of green, red, and blue. Instead, all colors appear to be dispersed throughout the network, indicating that some new entrants to the network quickly became central. However, you will note that there are no yellow nodes on the periphery of the network. In addition, earlier entrants seem to dominate the most densely connected section of this component of the network, suggesting that there may be a mild tenure effect.

In 2001 (Figure 5), some of the cliques apparent in 2000 merge into a more cohesive whole and with the growth of new entrants the network becomes more connected. The mild tenure effect becomes stronger in 2001 and 2002 (Figure 6) and we begin to see a

process of increasing concentration. Developers entering the network in 2001 (grey) and in 2002 (silver), concentrate on the periphery of the network. People who entered the network in its first two years (yellow and green nodes) dominate the center of the network. This suggests that new nodes may be attaching to nodes that are more central. However, there are several blue nodes that have only been a part of the network for one year, but are quite densely connected and quite central. Thus, while early incumbents dominate the center of the network, they do not bar energetic newcomers from entry.

Debian as a Bounded Entity. Prior research has shown how scientists engage in boundary-work practices to establish their epistemic authority (Gieryn, 1999) and draw rhetorical boundaries between real science and pseudo science (Gieryn, 1983). These types of boundary practices reinforce the jurisdiction of science and are not unlike the practices assumed by Debian members to secure authority over their project. However, Debian's concerns were far more pragmatic: securing their code and they had far fewer institutional supports to work with than the scientists studied by Gieryn (1983, 1999).

In terms of evaluating the efficacy of Debian's boundary practices, we conclude that Debian remains non-commercial, but the number of third parties that provide support and value added services has increased. Debian's development environment and source code remain publicly accessible, but becoming a developer is no longer as easy as it once was. Even the current Debian leader agreed that it was much harder to become a member now than when he first joined the project.

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 nowfor 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).

It can now take six months or more for a new maintainer to become a developer¹³.

Candidates now ask Debian developers to write a letter of reference for them.

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 overturned by a $\frac{2}{3}$ vote and an ultimate rejection cannot be over turned¹⁴. As of this writing, 645 members went through the new process, 133 applications were in progress, 29 candidates were awaiting a sponsor, and 33 applications were on hold¹⁵. Thirty-three application managers oversee the process. Fifty-nine people in 23 countries¹⁶ were looking to have their key signed despite the fact that 282 people in 39 countries offer to sign keys¹⁷.

The new membership process may reinforce network expansion through preferential attachment. If that is the case, new nodes will not enter randomly, but 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 highly connected and the bulk of nodes are only loosely connected. Since developers are more likely to attain leadership positions if they are more centrally connected, and since new members are likely to create ties with developers with higher centrality, this pattern of growth reinforces stability in gatekeeper

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.

¹⁴ Unfortunately, application rejection data was not available.

¹⁵ See <http://nm.debian.org> for more information.

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

¹⁷ See http://nm.debian.org/gpg_need.php for more information.

positions. As suggestive evidence of this pattern, the leader of the New Maintainer Committee was elected Project Leader the following year of our study.

Discussion

Despite the fact that some researchers have argued that open source projects do not rely on trust (Gallivan, 2001), prior research has shown that trust is critical for virtual teams (Jarvenpaa and Leidner, 1999) and this case is no exception. As Nohria and Eccles predicted (1992), “there may be a minimum ratio of face to face to electronically mediated exchange that is vital to maintain in order for a network organizations to work effectively” (1992: 290). Debian’s challenge was to protect against “trojans” while maintaining an open environment conducive to the norms and values of the open source community. The explosive unregulated growth of developers led project members to close their membership process and create a new governance mechanism that could help ensure that skills, goals, and ideology of new members were in line with that of the collective.

With this research, we uncovered structural factors that affected who became a gatekeeper for the project and thereby an architect of network expansion. Acquiring a central position in the network enhanced the probability of attaining a gatekeeper position even after controlling for technical contribution, ties with the project leader, tenure and geographic location. This is despite the fact that most researchers to open source projects claim that such projects are guided by purely meritocratic concerns. Our qualitative data shows that gatekeepers determined the new rules for membership, designing a membership process that requires sponsorship and a face-to-face meeting. By requiring sponsorship, the new maintainer process contributes to further centralization of the network. Dynamic analysis of the expansion of the network shows how a network becomes both more

connected and concentrated over time. This combination of research approaches illustrates how the dynamic evolution of a network affects the design of governance mechanisms that can, in turn, institutionalize social structures that foster preferential attachment.

This research also shows how a reinforcing cycle of preferential attachment helps sustain a scale invariant distribution. Previous research on the evolution of networks in the biotechnology industry (Powell et al, 2002; Owen-Smith et al, 2003) found that actors in this field had a preference for diversity as opposed to incumbents or like others (homophily). However Owen-Smith and colleagues (2003) found that while newcomers may be welcomed to the biotechnology network, an ‘open’ elite defines the standards of action for the field. Similarly, in this analysis of Debian’s social network, the elites that emerge define the terms of admittance. In doing so they help the project to stay open but bounded, and thereby sustain their own structural advantage.

Barabasi and Albert suggested that scale-free networks “are the inevitable consequence of self-organization due to the local decisions made by the individual vertices, based on information that is biased toward the more visible (richer) vertices, irrespective of the nature and origin of this visibility” (1999: 512). We argue that these processes are hardly inevitable and that in many real world social networks such bias can be manipulated by design. Here we provide some insight as to the origin and nature of structural advantage in a social system that exists primarily on-line, but requires real world authentication.

We have shown that the membership rules developed by Debian’s gatekeepers led to a progressive ‘funneling’ of the project’s social structure. This funneling process may be common to many types of open social systems that are challenged by unregulated growth. One way to take Lamont and Molnar’s call for greater integration in the study of boundaries seriously (2002) is to analyze how other types of social systems confront similar challenges.

Another way to further our understanding of common boundary processes across social systems is to examine how social networks map onto shifts in projected and enacted symbolic and social boundaries over time. Furthermore, what future research should determine is whether this type of funneling process can be reversed or whether, once in place, it continues to solidify and narrow points of access.

This research has broader implications for understanding knowledge communities that rely on novel governance mechanisms. First it suggests that literature on virtual or online communities remains naïve to the realities of an online production community. Online communities and virtual organizations are often depicted as nebulous and constituted by a shifting and ever changing body of members. Volatility and turnover in participation is not the same as indeterminacy. Flow is not the same as ambiguity. Organizational theorists who use terms such as porous and permeable organizational boundaries need to be more precise in this regard. Boundary definition and management will be critical for knowledge producing communities that wish to be open, but must do so without jeopardizing the security and stability of their work or its relationship with commercial enterprise.

David and Foray argue that the types of challenges faced by the Debian project are present in other types of distributed knowledge producing communities (2002).

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: increasing specialization, increasingly asymmetrical distribution of information and assessment capabilities, *ever-greater anonymity among interlocutors and ever-more opportunities for forgery of identity. Clearly new methods need to be devised to "Certify" the knowledge circulating on the Internet.....* (David and Foray, 2002: 17)

If distributed, pluralistic knowledge producing communities are to survive, innovations in reliability mechanisms will be critical to their sustenance. Research on mechanisms that help carve a protected, common informational space for collaboration is needed to understand

not only their emergence, but also their contribution to project security and effectiveness. While this research provides suggestive evidence that boundary management practices affect a social group's sustainability, further research linking boundary processes to outcomes is needed.

Such research should also attend to the legal platform required for such work to flourish as well as the design of the socio-technical space for collaboration. Prior to reaching the point where growth created new membership challenges for Debian, all contributors were assured that the work they created would remain non-proprietary and non-commercial¹⁸. The selection of people who share the project's philosophy is one way in which project boundaries with the commercial world become defined. However, intellectual property remains a constant source of concern for community managed projects like Debian. The second level of boundary management occurs at the code acceptance level. New challenges with regards to derivative works, proprietary modules and drivers often inspire division among members as to where the non-commercial boundary of Debian falls.

It is argued that in response to challenges to the norms of open science, well-defined communities of academic practice and credit are giving way to much larger co-owned intellectual property resource pools (Hellstrom, 2003). If academic research and commercial research are forging new areas of interdependence, particularly in the biomedical field, then scoping rights of access and belonging to emerging research opportunities will be a future challenge. David and Foray acknowledge as much in their prescient analysis of knowledge communities: "the potential for producing and reproducing knowledge will become greater as a community expands, but then so will the costs of data search, the risk of congestion and anonymity amongst members, which can in turn, represent a source of acute problems of

¹⁸ For more information on Debian's Social Contract with its members and community of users, refer to: http://www.debian.org/social_contract or O'Mahony (2004) for further analysis.

trust” (2002: 8). We would argue that the problems of trust described by David and Foray (2002) 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

- Albert, R. and A.-L. Barabási (2002). Statistical mechanics of complex networks,” *Review of Modern Physics* 74: 47.
- Barabasi, Albert-Laszlo and Reka Albert. (1999). Emergence of Scaling in Random Networks, *Science* 286: 509-512.
- Batagelj, V. and Mrvar, A: Pajek. (1998). A Program for Large Network Analysis. *Connections*, 21(2): 47-57.
- Brennen, Alex V. (2003). GnuPG Keysigning Party HOWTO. Located at: <http://www.cryptnet.net/fdp/crypto/gpg-party.html>.
- Coriat, Benjamin and Fabienne Orsi. (2002). Establishing a new intellectual property rights regime in the United States, *Research Policy* 31: 1491-1507.
- Dalle, Jean-Michel and David, Paul A. (2003). The Allocation of Software Development Resources in 'Open Source' Mode.
- Dasgupta, Partha and Paul A. David. (1994). Toward a new economics of science, *Research Policy* 23: 487-521.
- David, Paul A. (2000). The Digital Technology Boomerang: New Intellectual Property Rights Threaten Global “Open Science”, *forthcoming in World Bank Conference Volume,-2000*.
- David, Paul A. (2003). The Economic Logic of "Open Science" and the Balance between Private Property Rights and the Public Domain in Scientific Data and Information: A Primer. Stanford Institute for Economic Policy Research Discussion Paper No. 02-30.
- David, Paul A. and Dominique Foray. (2003). Economic Fundamentals of the Knowledge Society, *Policy Futures in Education*. An e-Journal, 1(1) Special Issue, January.
- Fleming L, Waguespack D. (2003) Merit, Status, or Networking? The Emergence of Leaders in a Technological Community. Harvard Business School Working Paper, Aug 28, 2003.
- Free Software Foundation. (1999) The GNU Privacy Handbook, located at: <http://www.gnupg.org/gph/en/manual/book1.html>.
- Gallivan, M. J. (2001). “Striking a Balance between Trust and Control in a Virtual Organization: A Content Analysis of Open Source Software Case Studies,” *Information Systems Journal* 11: 277-304.
- Gieryn T F. (1983). Boundary-work and the demarcation of science from non-science: strains and interests in professional interests of scientists, *American Sociological Review* (48): 781-95.

- Gieryn, T. (1999) *Cultural Boundaries of science: credibility on the line*, Chicago: University of Chicago Press.
- Hellstrom, Tomas. (2003). "Governing the virtual academic commons," *Research Policy* 32: 391-401.
- Jarvenpaa, S. and Leidner, D. (1999). "Communication and Trust in Global Virtual Teams," *Organization Science* 10: 791-815.
- Kamada, T. and S. Kawai, "An Algorithm for Drawing General Undirected Graphs", *Information Processing Letters* 31 (1989), 7-15.
- Kanter, Rosabeth M. (1968). Commitment and social organization: a study of commitment mechanisms in utopian communities. *American Sociological Review* 33: 499-517.
- Koch, S. and G. Schneider. (2000). *Results from Software Engineering Research into Open Source Development Projects Using Public Data..* Located at <http://opensource.mit.edu/papers/kochossoftwareengineering.pdf>
- Kogut, Bruce and Anca Metiu. (2001). Open-Source Software Development and Distributed Innovation, *Oxford Review of Economic Policy* 17: 248-264.
- Lamont, Michele & Molnar, Virga. 2002. The Study of Boundaries in the Social Sciences. *Annual Review of Sociology*, 28: 167-195.
- Latour, Bruno and Steve Woolgar. 1979. *Laboratory life: the social construction of scientific facts*. Beverly Hills: Sage Publications.
- Long, J. S. (1997). *Regression Models for Categorical and Limited Dependent Variables*. Thousand Oaks, CA: Sage.
- Merton, Robert K. (1973). *The Sociology of Science*. Edited by Norman W. Storer. Chicago, IL: University of Chicago Press.
- Michlmayer M, & Mako Hill B. "Quality and the Reliance on Individuals in Free Software Projects" Presented at *Taking Stock of the Bazaar: 3rd Workshop on OS SW Eng*, Portland OR, May 3-10, 2003. Located at: <http://opensource.ucc.ie/icse2003/>.
- Mockus A., Fielding, R.T, Herbsleb, J. (2000). *A Case Study of Open Source Software Development: The Apache Server*, Proceedings of ICSE.
- Mockus, Audris, Roy T. Fielding, and James D. Herbsleb. (2002). Two Case Studies of Open Source Software Development: Apache and Mozilla, *ACM Transactions on Software Engineering and Methodology* 11 (3): 309-346.
- Mowery, David C. and Bhaven Sampat. (2004). The Bayh Dole Act of 1980 and University Industry Technology Transfer: A Model for Other OECD Governments? In *'Ivory Tower' and Industrial Innovation: University-Industry Technology Transfer Before and After the Bayh-Dole Act*. Palo Alto: Stanford University Press.

- Mowery, David C. and Arvids A. Ziedonis. (2002). Academic patent quality and quantity before and after the Bayh-Dole act in the United States, *Research Policy* 31: 399.
- Mowery, David C. and Bhaven N. Sampat. (2001). "Patenting and licensing university inventions: Lessons from the history of the research corporation," *Industrial and Corporate Change* 10: 317.
- Murdock, I. "A Brief History of Debian". Appendix A, The Debian Manifesto, Revised January 6, 1994, located at: <http://www.debian.org/doc/manuals/project-history/ap-manifesto.en.html>.
- Network Associates. (1990). An Introduction to Cryptography, located at: <http://www.pgpi.org/doc/guide/6.5/en/intro/>.
- Nohria, Nitin and Robert G. Eccles. (1992). Face-to-Face: Making Network Organizations Work. In *Networks and organizations structure, form, and action*. Edited by Nitin Nohria and Robert G Eccles. Boston, Mass: Harvard Business School Press.
- O'Mahony, Siobhan. (2002). Community Managed Software Projects: The Emergence of a New Commercial Actor, unpublished dissertation, Stanford University.
- O'Mahony, Siobhan. (2003). Guarding the Commons: how community managed projects protect their work, *Research Policy* (32): 1179-1198.
- O'Mahony, Siobhan. (2004). Managing Community Software in a Commodity World, to appear in *Ethnographic Reflections on the New Economy*, (Eds.) Melissa Fisher, and Greg Downey, Duke University Publications.
- Owen-Smith, Jason (2005) "Dockets, Deals, and Sagas: Commensuration and the Rationalization of Experience in University Licensing. Forthcoming. *Social Studies of Science*.
- Owen-Smith, Jason, Walter W. Powell, and Kenneth W. Koput. (2004). Pathways to an Open Elite in the Life Sciences. In *Market Emergence and Transformation*. Edited by J. Padgett and W. Powell.
- Owen-Smith, Jason. (2003). From separate systems to a hybrid order: accumulative advantage across public and private science at Research One universities, *Research Policy* 32: 1081-1104.
- Owen-Smith, Jason and Walter W. Powell. (2003). The expanding role of university patenting in the life sciences: assessing the importance of experience and connectivity, forthcoming, *Research Policy*.
- Powell, Walter W. and Jason Owen-Smith. (1998). Universities and the market for intellectual property in the life sciences, *Journal of Policy Analysis and Management* 17 (2): 253-277.
- Powell, Walter W, Koput, Kenneth W, and Smith-Doerr, Laurel. (1996). Interorganizational collaboration and the locus of innovation networks of learning in biotechnology.

Administrative Science Quarterly 41: 116-145.

Powell, Walter W. (1990). Neither Market nor Hierarchy: Network Forms of Organization. Staw, Barry M. and Cummings L. L. *Research in Organizational Behavior* 12: 295-336.

Raymond, Eric S. (1999). *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. Sebastopol, CA: O'Reilly & Associates.

Rothschild, Joyce and Raymond Russell. (1986). "Alternatives To Bureaucracy: Democratic Participation in the Economy," *Annual Review of Sociology* 12: 307-328.

Rothschild, Joyce and J. Allen Whitt. (1986). *The Cooperative Workplace: Potentials and dilemmas of organizational democracy and participation*. New York: Cambridge University Press.

Rothschild-Whitt, Joyce. (1979). The Collectivist Organization: An Alternative to Rational-Bureaucratic Models. *American Sociological Review* 44: 509-527.

Stallman, Richard. 1999. The GNU Operating System and the Free Software Movement. In *Open Sources*. Edited by Chris DiBona, Sam Ockman, and Mark Stone. Sebastopol, CA: O'Reilly.

Tuomi, Ilkka. (2003). *Networks of Innovation: Change and Meaning in the Age of the Internet*. Oxford: Oxford Press.

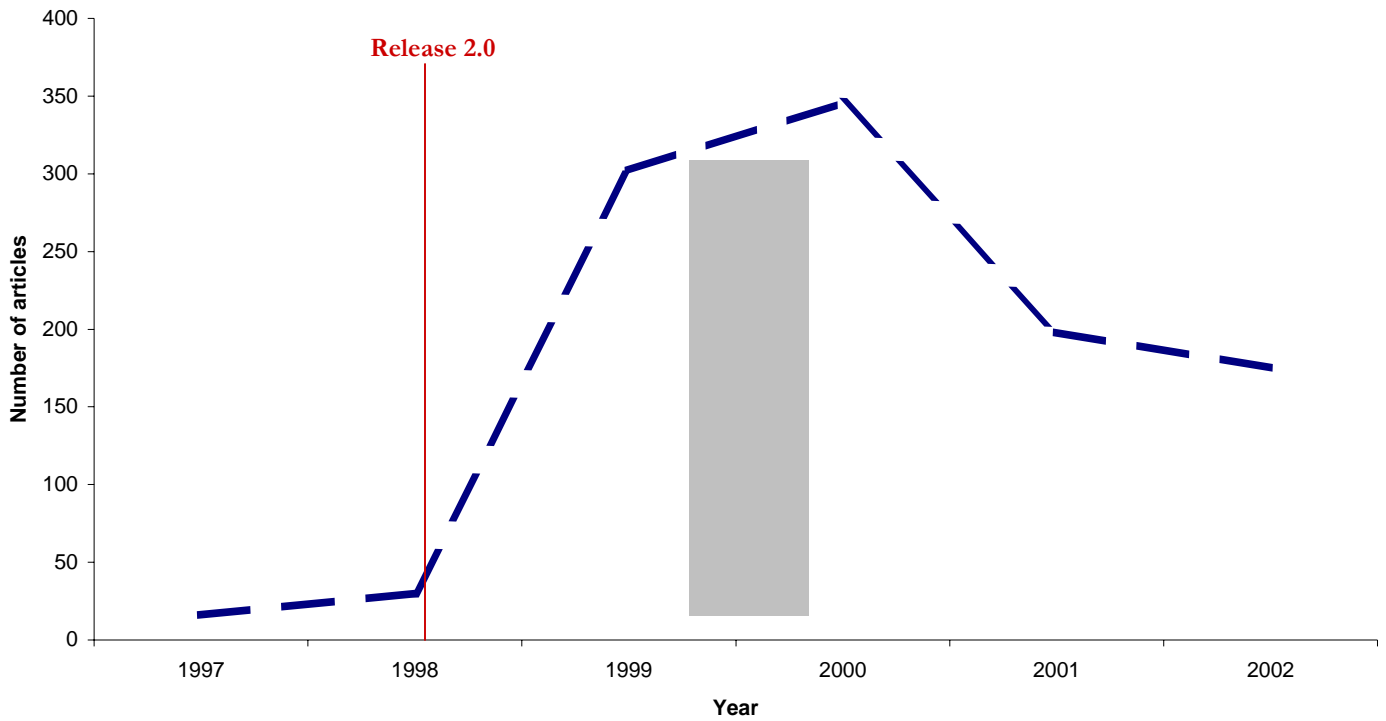
Ulrich, Bill. Debian GNU/Linux 2.2 (Potato), *PC Magazine*, November 13, 2001, located at <http://www.pcmag.com/article2/0,4149,16093,00.asp>.

Von Krogh, Georg, Sebastian Spaeth, and Karim R. Lakhani. (2003). Community, Joining, and Specialization in Open Source Software Innovation: A Case Study, *Research Policy forthcoming*.

Wayner, Peter. 2000. *Free for All: How Linux and the Free Software Movement Undercut the High-Tech Titans*. New York, NY: HarperCollins.

Whyte, William Foote and Kathleen King Whyte. (1988). *Making Mondragon: The Growth and Dynamics of the Worker Cooperative Complex*. Ithaca, NY: ILR Press: New York State School of Industrial Labor Relations, Cornell University.

Figure 1: Media Citations of Debian GNU/Linux 1997 - 2002



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.

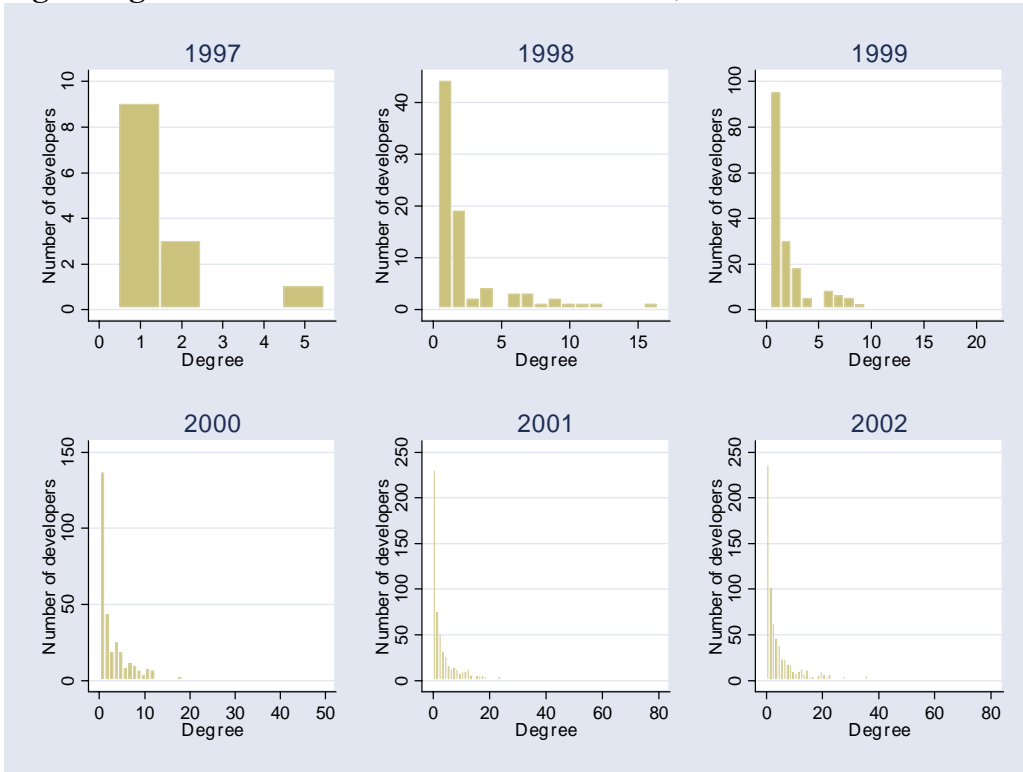
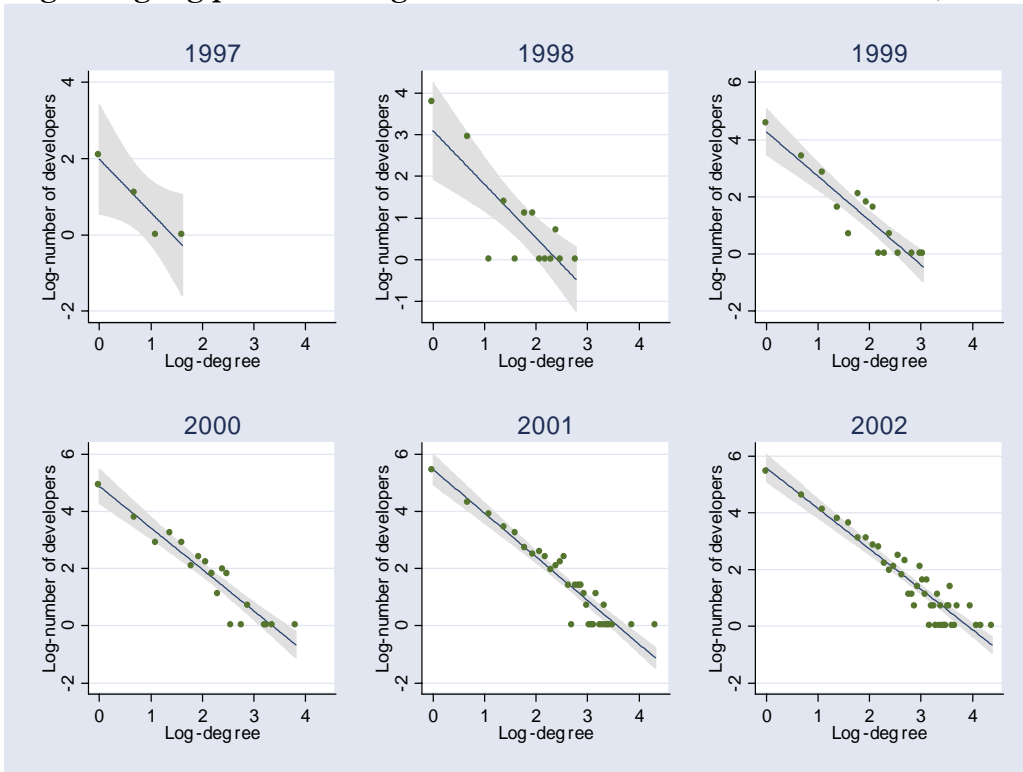


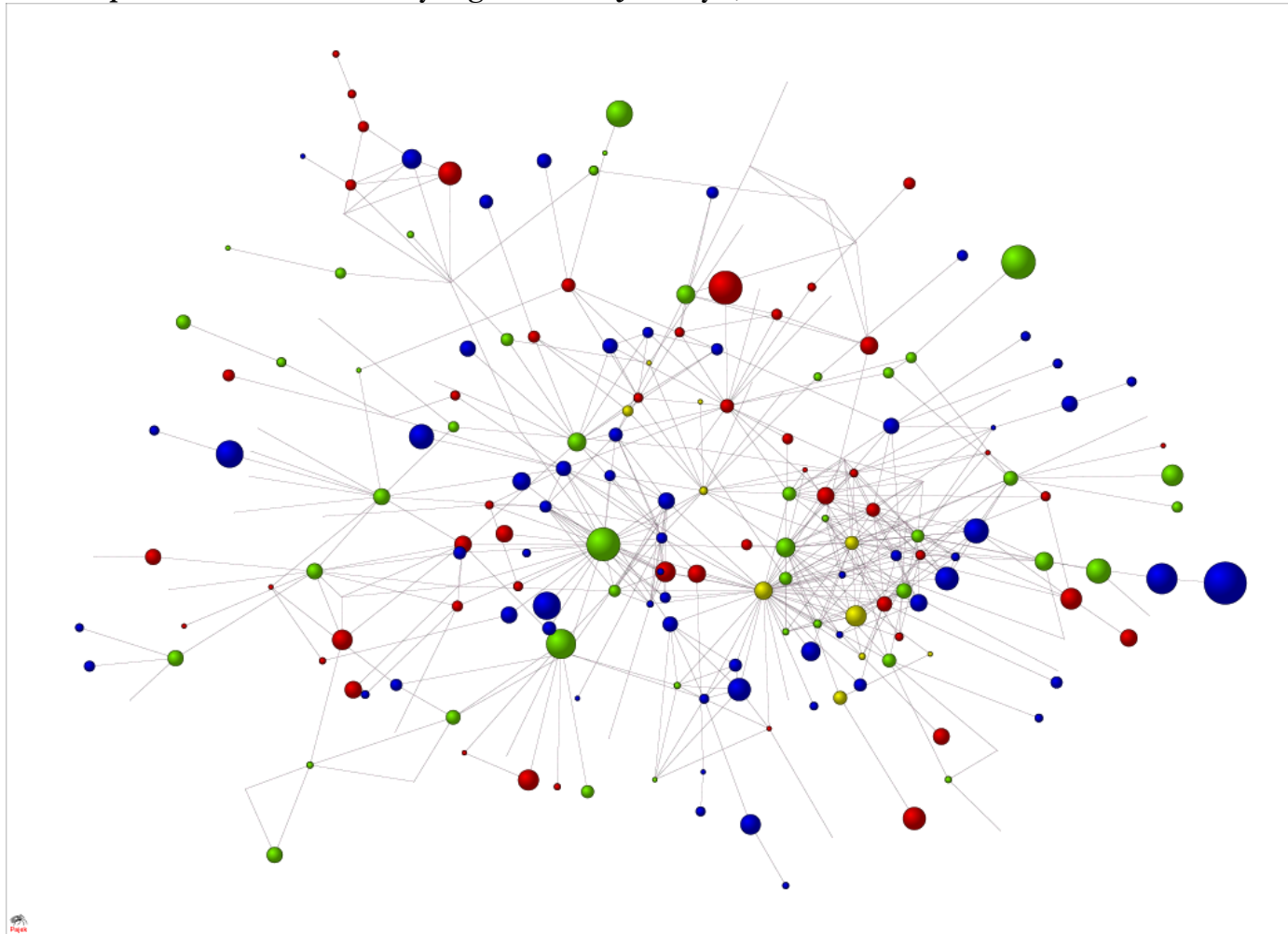
Fig. 3 Log-log plot of the degree distribution of the Debian Network, 1997-2002.



Note:: The graphs report the scatter plot of the log-log degree distribution and a regression line, with the shaded area representing the 95% confidence interval around the predicted value (the straight line that we would expect to find according to the scale-free model).

Figure 4: Main Component of the Debian Keyring Network: January 1, 2000

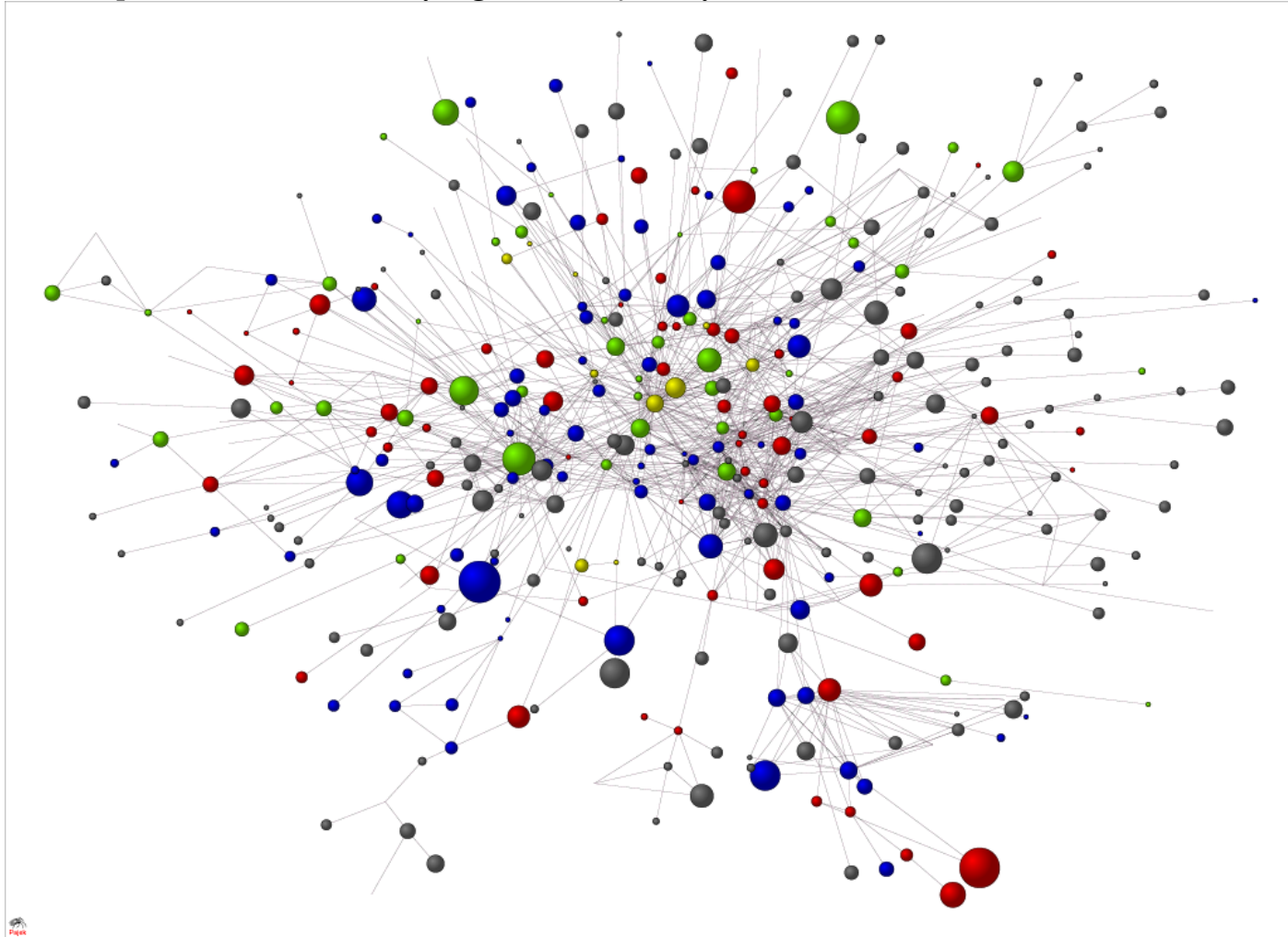
Nodes = 212



Note: In this visualization of the main component of the social network of Debian Developers, the color of the nodes indicates the year the developer joined the community. Gold = 1996 or earlier; green = 1997; red = 1998; blue = 1999. The size of the node measures the number of package maintained by the developer.

Figure 5: Main Component of the Debian Keyring Network: January 1, 2001

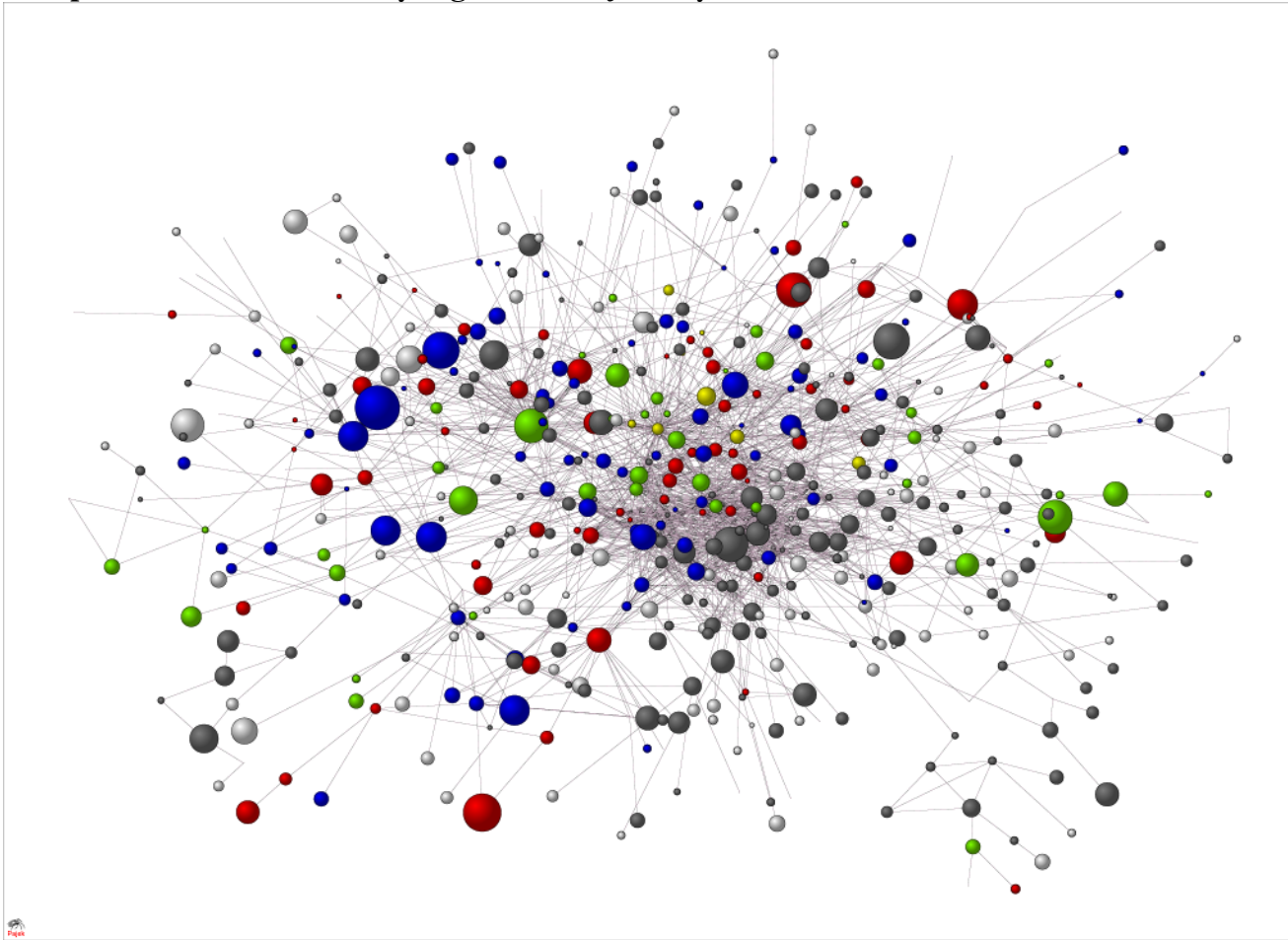
Nodes = 427



Note: In this visualization of the main component of the social network of Debian Developers, the color of the nodes indicates the year the developer joined the community. Gold = 1996 or earlier; green = 1997; red = 1998; blue = 1999; grey = 2000. The size of the node measures the number of package maintained by the developer.

Figure 6: Main Component of the Debian Keyring Network: January 1, 2002

Nodes = 592



Note: In this visualization of the main component of the social network of Debian Developers, the color of the nodes indicates the year the developer joined the community. Gold = 1996 or earlier; green = 1997; red = 1998; blue = 1999; grey = 2000; White = 2001. The size of the node measures the number of package maintained by the developer.

Table 1: Growth in the Debian Keyring Network

	1997	1998	1999	2000	2001	2002^a
Number of Developers in network	13	82	176	298	532	671
Growth rate	-	530.77%	114.63%	69.32%	79.32%	26.13%
Number of ties	11	111	239	543	1212	2014
Average Degree	2.46	2.71	2.72	3.64	4.56	6.00
S.D. Degree	1.60	3.04	3.22	4.67	6.57	8.84
Number of Components (min 2)	3	19	29	31	39	33
Density	0.124	0.033	0.015	0.012	0.009	0.009

^a Based on the developers' network on sep. 22, 2001

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

Dependent Variable	2001		2002	
	All Developers (N=532)		All Developers (N=671)	
	Mean	S.D.	Mean	S.D.
New Maintaner Committee ^a	.010		.005	
Independent Variables				
Number of Packages maintained (as of 1/1/year t)	6.68	9.01	7.23	9.87
Package popularity (2003)	299.5	809.8	270.9	749.6
Tenure (in months) (year t-1)	18.54	15.50	22.89	16.41
Europe (reference category) ^a	.46		.47	
North-America ^a	.31		.31	
Other continent ^a	.12		.13	
Tie to Leader ^a (year t-1)	.11		.12	
# of mailing list postings (year t –1)	12.24	28.43	13.54	42.35
Degree Centrality	4.56	6.58	6	8.85

^aDummy variables

Table 3: Correlation coefficients in 2001.

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

Table 4: Correlation coefficients in 2002.

Variables	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
(1) New Maintainer committee	-								
(2) # of packages maintained	0.0846	-							
(3) Package popularity	0.0751	0.3738	-						
(4) Degree centrality	0.0725	0.0796	0.0892	-					
(5) Tie to Leader	0.0076	0.0177	0.0982	0.3513	-				
(6) # of mailing list postings	0.0641	0.2208	0.3752	0.2787	0.0410	-			
(7) Tenure (months)	-	-	0.1552	0.1805	0.2396	0.0430	-		
(8) North-America	0.0568	-	0.0180	-	-	-	-	-	
(9) Others	-	0.0804	0.0361	-	-	0.0608	-	-	-
	0.0422			0.1035	0.0752		0.0647	0.2630	

Table 5: Logistic Regression coefficients for the regression of New Maintainer Committee membership on selected independent variables in 2001.

	(1)	(2)	(3)
Number of Packages	0.035** (1.036)	0.033** (1.034)	0.033** (1.033)
Package Popularity	0.0002 (1.0002)	0.0003 (1.0003)	0.0004* (1.0004)
Tenure in the Project	-0.027** (0.974)	-0.028** (0.973)	-0.049*** (0.952)
Tie to leader (year t-1)	1.194*** (3.299)	1.166*** (3.211)	0.170 (1.186)
North America ^a	-0.095 (0.910)	-0.133 (0.876)	0.153 (1.165)
Other Continent ^a	-0.139 (0.870)	-0.152 (0.859)	0.138 (1.147)
Number of Postings (year t-1)		0.003 (1.003)	-0.001 (0.999)
Degree Centrality			0.142*** (1.152)
Intercept	-2.292***	-2.303***	-2.686***
Log-likelihood ratio for model estimated:			
vs. null model (df)	23.73†† (6)	24.89†† (7)	57.63†† (8)
vs. previous model (df)		1.16 (1)	32.74†† (1)
Observations	515	515	515
Pseudo R-squared	0.07	0.07	0.17

Odds ratios in parentheses

^a Compared to developers located in Europe

*=p<0.1, **=p<0.05, ***=p<0.01 (one tailed tests)

† χ^2 significant at the level (p = <.05)

†† χ^2 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

	(1)	(2)	(3)
Number of packages	0.025 (1.025)	0.022 (1.023)	0.024 (1.024)
Package Popularity	0.0004* (1.0004)	0.0003* (1.0003)	0.0004* (1.0004)
Tenure in the Project	-0.060*** (0.942)	-0.060*** (0.942)	-0.069*** (0.933)
Tie to leader (year t-1)	0.662 (1.939)	0.673 (1.959)	0.381 (1.463)
North America ^a	0.397 (1.487)	0.398 (1.489)	0.616 (1.852)
Other Continent ^a	-0.799 (0.450)	-0.817 (0.442)	-0.556 (0.573)
Number of Postings (year t-1)		0.002 (1.002)	0.001 (1.001)
Degree Centrality			0.044** (1.045)
Intercept	-2.442***	-2.481***	-2.717***
Log-likelihood ratio for model estimated:			
vs. null model (df)	21.52†† (6)	22.31†† (7)	27.11†† (8)
vs. previous model (df)		2.37 (1)	4.8† (1)
Observations	647	647	647
Pseudo R-squared	0.09	0.09	0.11

Odds ratios in parentheses

^a Compared to developers located in Europe

*=p<0.1, **=p<0.05, ***=p<0.01 (one tailed tests)

† χ^2 significant at the level (p = <.05)

†† χ^2 significant at the level (p = <.01)