The Ecology of Open Source Software Development

Kieran Healy

University of Arizona

kjhealy@arizona.edu

Alan Schussman

University of Arizona

schussat@email.arizona.edu

Open Source Software (OSS) is an innovative method of developing software applications that has been very successful over the past eight to ten years. A number of theories have emerged to explain its success, mainly from economics and law. We analyze a very large sample of OSS projects and find striking patterns in the overall structure of the development community. The distribution of projects on a range of activity measures is spectacularly skewed, with only a relatively tiny number of projects showing evidence of the strong collaborative activity which is supposed to characterize OSS. Our findings are consistent with prior, smaller-scale empirical research. We argue that these findings pose problems for the dominant accounts of OSS. We suggest that the gulf between active and inactive projects may be explained by social-structural features of the community which have received little attention in the existing literature. We suggest some hypotheses that might better predict the observed ecology of projects.

Besides the Internet itself, the most distinctive and widespread development in information technology since the mid-1990s has not been a particular software application, but rather a way of writing and distributing software in general. The rapid growth of the Open Source Software (OSS) movement has astonished most observers. From modest beginnings, there are now thousands of active OSS projects, ranging in scope from simple calculators to entire operating systems. Open Source Software is given away for free by the developers who write it, both in the sense that it is provided

at a nominal charge and that it is licensed to users without the legal restrictions typical of commercial software.[1]

Although OSS development is an important phenomenon closely linked to the growth of the Internet, we do not know much about how it works in practice, who is involved or why they participate. From a theoretical point of view, the OSS approach does not fit with standard models of software development (Sandred 2001; Vixie 1999) or formal organization more generally (Neff and Stark 2002). It is a hybrid: part social movement, with idealistic principles and goals; part formal organization, with an intensive schedule and innovative products; part volunteer network, with time and energy to donate.

In this paper, we present data from a large sample of OSS projects as a first step towards understanding the social organization of OSS development. We argue that, as a collaborative project of great size and scope, rooted in networks of volunteers and embedding innovative organizations and new markets within itself, OSS development is of strong interest to economic sociologists. We find that the social structure of OSS development, as measured by the size, composition and activity pattern of projects, differs significantly both from typical characterizations by movement "evangelists" and from broad claims by theorists of OSS.

We begin by providing some basic background about OSS and then discuss the emerging theoretical literature. We then present some hypotheses from this literature and test them against a large sample of OSS projects. A central finding is that the OSS community is spectacularly stratified on a variety of activity and participation measures. The observed structure challenges both the common image of the OSS community as a relatively "flat" network of interacting peers (Raymond 2001) and to an emerging theory of the community as an efficient router of human capital information (Benkler 2002). It *is* consistent, however, with some recent empirical data on OSS project development (Mockus, Fielding, and Herbsleb 2000) and on the OSS community (Krishnamurthy 2002). In our discussion, we emphasize the strikingly consistent patterns found in the data, discuss their implications for existing theory, and suggest a number of hypotheses about the social structure of the OSS community that might explain the observed patterns.

EXISTING THEORY AND RESEARCH

Traditional software development follows a familiar pattern. A company writes a program and then tries to sell it. The program's "source code" is a trade secret, just like

[1]For an accessible narrative overview of the OSS movement, including samples of the various software licenses, see Wershler-Henry 2002. For some of the foundational texts of the movement, and a sense of its internal diversity, see the essays in DiBona, Ockman, and Stone 1999.

an auto manufacturer's blueprints for its cars. By contrast — as the name suggests — the source code in an OSS project is available to anyone who wants it. The software is developed and maintained by a community of volunteers. Essentially, anyone is free to take the results of this work and modify it, extend its capabilities, or incorporate it into their own projects (usually, but not always, on condition that they keep their own contributions "open" in a similar way).[2] It is a counterintuitive way to act in the highly competitive world of computing and information technology. And yet, although casual users are often unaware of it, many essential features of the Internet — such as sending email and serving web pages — are more often than not controlled by software created through OSS projects. Besides this "backstage" work, OSS projects like the GNU/Linux operating system and, more recently, Apple's use of OSS components in their operating system have become increasingly prominent.

Early surveys of the new information technologies were largely speculative and tended to focus on the transformative possibilities of new hardware (Dertouzos 1997; Gilder 1990). More recent research has put the growth of the Internet in historical perspective (Abbate 2000; Chandler and Cortada 2000) and shown how the new information technologies are changing — and being shaped by — existing legal and political institutions (Lessig 2001; Sunstein 2001). Sociologists are also examining the effect of the Internet on social inequality, political participation and cultural life generally (for a review see DiMaggio et al. 2001).

A smaller body of research focuses on the engine driving many of these changes — the OSS developers themselves and the distinctive organizational forms they have created. Evangelists for the movement have written about how it ought to work (DiBona, Ockman, and Stone 1999), and journalists have described the movement's recent history and main figures (Moody 2001). But there are far fewer studies of the organizational structure of OSS, such as the characteristics of projects, their practical administration, the relationship between developers and users, the location and background of the developers themselves, their reasons for participating in the community, the benefits they gain from it, and so on.

---

[2] The precise conditions depend on the license the software is released under. There is debate within the OSS community about its appropriate form. The central innovation of the Free Software Foundation's canonical GNU General Public License (or GPL) is that it allows the licensee to make changes to the source code and freely redistribute it only on the condition that the source code for the new version is redistributed with any new version of the software. Programmers may freely use the work of others, but must make their own changes available for use in the same way. Other licenses (the Berkeley BSD license, for example) do not contain this restriction. See Sandred 2001 Ch. 3, Wershler-Henry 2002 Chs 2 and 4, Perens 1999 and Williams 2002 for further detail and different perspectives on the issues at stake.

Theories of oss

The stylized image of oss development, most often found in popular accounts of the phenomenon, is one of an egalitarian network of developers largely free of hierarchical organization and centralized control. A widely-cited essay by Eric Raymond argues explicitly that the virtues of the oss style of development come from its flat structure. He contrasts

> two fundamentally different development styles, the "cathedral" model of most of the commercial world versus the "bazaar" model of the Linux world... Linus Torvalds's style of development — release early and often, delegate everything you can, be open to the point of promiscuity — came as a surprise. No quiet, reverent cathedral-building here — rather, the Linux community seemed to resemble a great babbling bazaar of differing agendas and approaches ... out of which a coherent and stable system could seemingly emerge only by a succession of miracles ... the Linux world not only didn't fly apart in confusion but seemed to go from strength to strength at a speed barely imaginable to cathedral-builders (Raymond 1998).

Efforts by economists to "theorise the the bazaar," as it were, break the problem into two parts. First is the lack of hierarchy and the apparently chaotic nature of oss development. From an economist's point of view, it is not surprising that a distributed and disaggregated system of innovation successfully outperforms a hierarchically-organized alternative. Thus, Eric von Hippel argues that

> Complete user-centric innovation development and consumption communities can flourish when (1) at least some users have sufficient incentive to innovate, (2) at least some users have an incentive to voluntarily reveal their innovations and the means to do so, and (3) diffusion of innovations by users can compete with commercial production and distribution. When only the first two conditions hold, a pattern of user innovation and trial will occur, followed by commercial manufacture and distribution of innovations that prove to be of general interest (Hippel, n.d.).

By "user-centric innovation" von Hippel means a community of users who tend to tinker with and improve the products they use. Hobbyists often display this tendency (von Hippel first noticed it happening amongst windsurfers who improved their gear) and he argues that the oss community is a very large and disaggregated example of this type.

A complementary account is presented in Benkler 2002. Benkler presents a transactions costs analysis (**Williamson1985**; Coase 1988). The central virtue of the oss

model, he argues, is its "peer-to-peer" structure, which lowers transactions costs in particular circumstances and provides a more efficient method of allocating human capital inputs:

> [T]he primary advantage of peer production is in acquiring and processing information about human capital available to contribute to information production projects … [P]eer production has a systematic advantage over markets and firms in matching the best available human capital to the best available information inputs to create the most desired information products … [I]f peer production has a sufficient advantage in terms of its capacity to process information about who the best person is for a given information production job over firm and market based mechanisms to outweigh the costs of coordination, then peer production will outperform firms and markets (Benkler 2002).

Benkler does not say how to measure the benefits of information flow conferred by commons-based peer production. Information is not conveyed via price signals, as in markets, or via chains of command, as in firms. Instead, it "relies on decentralized information gathering and exchange to reduce the uncertainty of participants, and has particular advantages as an information process for identifying human creativity available to work on information and cultural resources in the pursuit of projects, and as an allocation process for allocating that creative effort" (7). The "pervasively networked environment" of the oss community allows for better information flow about who should take on particular projects. This analysis suggests a community social structure of a particular kind. In contrast to atomized agents relying on price signals to make decisions, community participants should be well-connected ("pervasively networked") and information should flow well between them. The result, according to Benkler, will be a process "matching human capital to information inputs to produce new information goods" (73).

The second problem is a little more difficult to reconcile with a purely economic view. Why is there so much voluntary participation? Why are participants willing to develop and distribute their innovations for free? In a careful economic analysis of developer motivations, Lerner and Tirole 2000 argue that, despite initial appearances, several incentives make it rational to volunteer. Chief amongst these are the practical benefits to users of having software that works properly, the increase in reputation that comes from being associated with a successful project, and the potential for oss projects to lead to further commercial opportunities. Lerner and Tirole argue that "the reputational benefits that accrue from successful contributions to open source projects appear to have real effects on the developers," that "there also appear be quite tangible — if delayed — rewards" to participation, that "many of the skilled Apache

programmers have benefited materially from their association with the organization" and that there is "substantial evidence that open source work may be a stepping stone to securing venture capital" (Lerner and Tirole 2002, 217-18).

This perspective on oss development is illuminating but incomplete. The motives adduced by Lerner and Tirole are plausible (and supported by evidence from a number of important oss projects), but cannot fully account for the movement's organization. In particular, the role of the project leader is difficult to explain in economistic terms:

> Another important determinant of project success appears to be the nature of its leadership … The key to a successful leadership is the programmers' trust in the leadership: that is, they must believe that the leader's objectives are sufficiently congruent with theirs and not polluted by ego-driven, commerical or political biases. In the end, the leader's recommendations are only meant to convey her information to the community of participants. The recommendations receive support from the community only if the leadership's goals are believed to be aligned with the programmers' interests (221-22).

The difficulties of the argument are apparent. Programmers are supposed to contribute to projects for self-interested reasons, but leaders can only be successful if they are "not polluted by ego-driven, commercial or political biases". Lerner and Tirole attempt to parse the project leader's authority as simply a matter of conveying information, and her success as simply dependent on congruence with the interests of her followers. But, from a sociological point of view, this is a thin characterization of the role of the charismatic leader. Indeed, they note that "leaders of open source movements may initially not have been motivated by ego gratification and career concerns" (213), and that "Despite the substantial status and career-concerns benefits of being a leader of an imporant open source project, it would seem that most should not resist the large monetary gains from taking a promising technology private. We can only conjecture why this is not the case" (215).

These gaps in the economic account of community participation suggest there is more than "simple economics" at work here. In particular, the key role of project leaders in mobilizing successful projects seems under-theorized. Empirical studies of the oss community's social structure raise further questions about both Lerner et al.'s and Benkler's view of oss.

Existing empirical data

What do we know about oss community structure? We focus on two studies in particular.[3] Mockus, Fielding, and Herbsleb 2000 is a case study of one of the most successful open source projects, the Apache server.[4] Taking a software-engineering perspective, they focused on the internal process used to develop Apache, asking how many people wrote code, reported problems and repaired defects. A key finding was that the Apache project was driven by about 15 core developers, who contributed about 85-90% of the code, surrounded by a larger penumbra of participants. A group larger than the core by about an order of magnitude helped fix defects, and a group larger by an order of magnitude again helped report problems (271). The authors hypothesized that this pattern would be common across successful oss projects.

In one of the few survey- rather than case-study based analyses of oss, Krishnamurthy 2002 asked whether "the community-based model of product development holds as a general descriptor of the average oss product." Examining 100 mature projects from the Sourceforge database (see below for further discussion of this data source, also used in this paper), he found that "the vast majority of mature oss projects are developed by a small number of individuals. In fact, the median number of developers in his sample was one. He also found that few oss projects generated much discussion. "On average, each oss project had 2 forums and 2 mailing lists for discussion. Ten of the 100 products had neither an online forum nor a mailing list ... 33 out of 100 projects had 0 messages! At the same time, a few products led to great discussion with the highest number of messages over a life time of a product standing at 4,952" (Krishnamurthy 2002).

These findings suggest that a large-scale survey of oss projects should find a clear structure. On the basis of the Apache case study, we expect that *within projects*, development activity will be strongly skewed across coding, problem-correction and problem-reporting tasks in, the manner described by Mockus, Fielding, and Herbsleb (2000). On the basis of Krishnamurthy's survey, we expect that *between projects*, development activity will be strongly skewed, with a small number of projects attracting most activity.

---

[3]Other important efforts in this direction include Lakhani and Hippel, n.d., Lancashire 2002 and Schweik and Semenov 2003.

[4]Apache is a piece of software that serves up web pages. Websites are powered by web servers that respond to requests from users to see a particular page on a site. More than half the world's websites are powered by the Apache server.

DATA AND METHODS

One reason for the lack of research in this area is that the oss movement is large, informally structured, and geographically dispersed. It is therefore difficult to collect good data. We avoid this problem by using a copy of the Sourceforge project database in our analysis. Sourceforge (www.sourceforge.net) is the largest repository of Open Source projects available on the Internet. Sourceforge provides hosting services for project developers, allowing them to manage their source code, communicate with one another (via email, mailing lists and discussion forums) and make their work available for download. Although the developers themselves are located all over the world, Sourceforge's servers are where the day-to-day interaction and innovation actually happens for a very large sample of oss projects.

Sourceforge's parent company, osdn, provided the lead author with a summary snapshot of its project database in August of 2002. The data contain records for the 46,356 projects then hosted by Sourceforge. Fields include basic (non-confidential) information on each project and various measures of development activity. These measures include data on the number of developers, downloads, cvs[5] checkouts and commits, number of mailing lists, number of posts to mailing lists, number of unique message authors, number of site views, number of support requests open and closed, and number of bug reports opened and closed. These variables measure different aspects of a project's vitality ranging from level of general interest (site views, downloads) to intensity of development (number of developers, cvs code commits), as well as intermediate activity (bug reports, support requests).

Sourceforge does not host all oss projects by any means. Several major projects — amongst others, the Linux Kernel, the Apache server, the GNOME and KDE desktop environments, and the XFree86 implementation of the X-Window system are all major oss projects with their own websites, project-management groups and underlying organizational apparatus (usually some form of non-profit organization). In addition, many of the original GNU software projects are hosted by the Free Software Foundation's Savannah server rather than by Sourceforge. Nevertheless, Sourceforge is a valuable resource for researchers. It is by far the largest collection of oss projects and has grown rapidly since its foundation in 1999, as can be seen from Figure 1.

---

[5]Concurrent Versioning System, or cvs, is software that allows multiple developers to work on a project over time. Developers "check out" portions of the source code in order to make modifications to it. Changes are later "committed" to the main development tree, which holds the canonical version of the code. The cvs application keeps track of what code has been checked out, and by whom, and allows project managers to reconcile code changes made by different developers into a stable release of the source code. Data on cvs checkouts and commits is therefore a measure of a project's development activity.
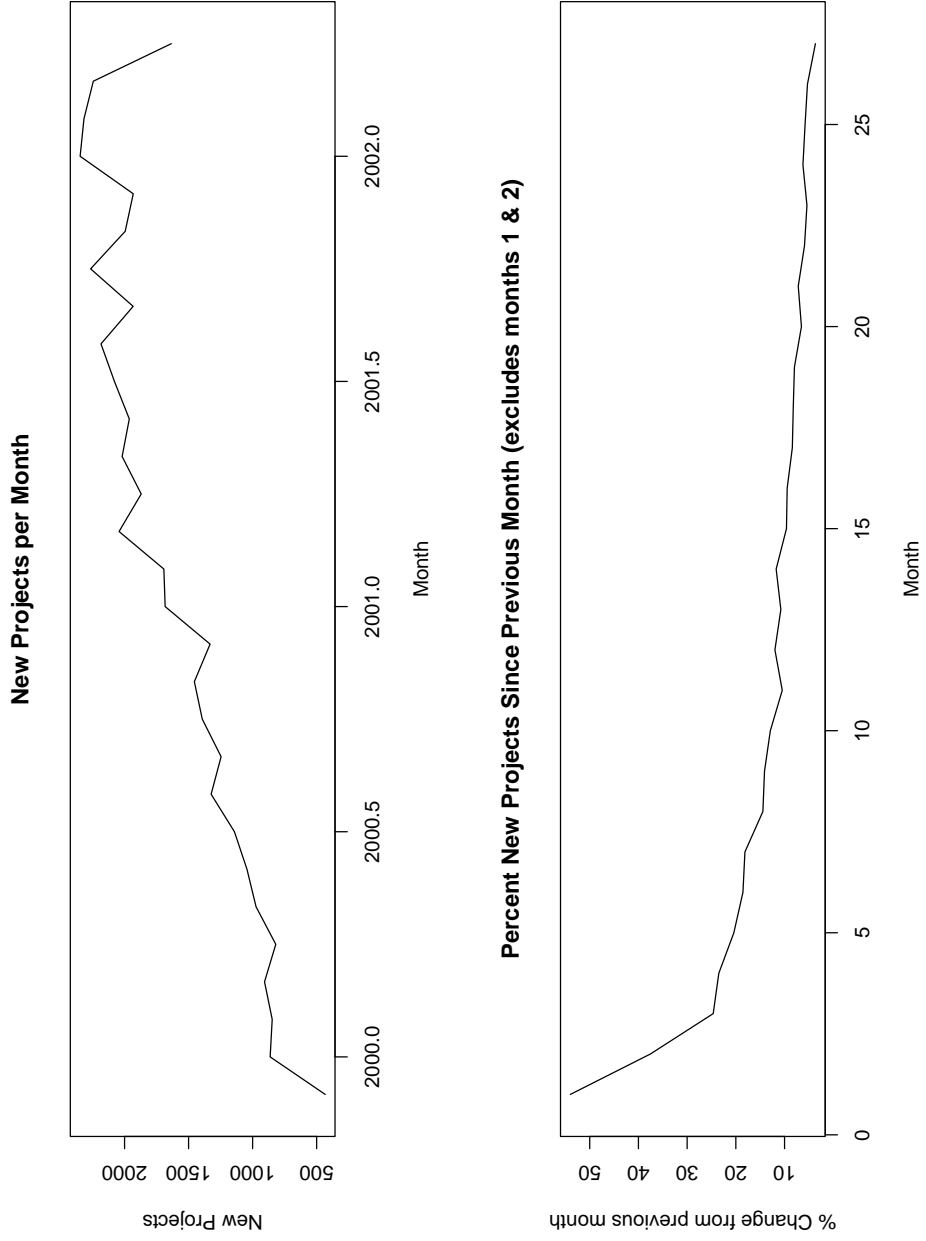
Figure 1: New Projects on Sourceforge, Nov. 1999–Aug 2002: absolute (top) and relative (bottom) growth.

RESULTS

We examined six measures of project activity. Detailed descriptive summaries of each measure are provided in Table 1.[6] It shows the valid N, the number of missing values and the number of unique values for each variable. Values at selected intervals between the 5th and 95th percentiles are shown beneath a small histogram. Each variable's five lowest and five highest values are also shown.

Table 1: Summary of Project Activity Measures

Develops

```
      n missing  unique   Mean    .05   .10   .25   .50   .75   .90    .95
  46356      0      43   1.688      1     1     1     1     2     3     5

lowest :  1  2  3  4  5, highest: 46 47 55 65 80
```

Downloads

```
      n missing  unique   Mean    .05   .10   .25   .50   .75   .90    .95
  46356      0    4380    2289      0     0     0     0    48   872   3116

lowest :      0      1      2      3      4
highest: 1698729 1723212 3330871 4598807 7703001
```

Site Views

```
      n missing  unique   Mean    .05   .10   .25   .50   .75   .90    .95
  46356      0    5726    1915      7    12    31   140   562  2084   4903

lowest :      0      1      2      3      4
highest:  640248  762994  821937  879960 5124577
```

Msg Uniq Auth

```
      n missing  unique   Mean    .05   .10   .25   .50   .75   .90   .95
  46356      0      70   1.237      0     0     1     1     1     2     3

lowest :  0  1  2  3  4, highest: 132 149 183 257 344
```

CVS Commits

```
      n missing  unique   Mean    .05   .10   .25   .50   .75   .90    .95
  46356      0    1649   88.89    0.0   0.0   0.0   0.0   1.0  94.0  328.2

lowest :     0     1     2     3     4
highest: 30111 31058 39669 68320 78959
```

---

[6]This table was produced using the describe function from Frank Harrell's Hmisc library for R (Harrell 2001; Ihaka and Gentleman 1996).

As is clear from these descriptive statistics, activity across the whole range of OSS projects is spectacularly skewed, beyond even the expectations raised by Krishnamurthy's study. The median number of developers is one. The 95th percentile is 5. Relative to the whole field, only a tiny number of projects have more than a handful of developers. The median number of CVS commits is zero. At the 75th percentile it is 1 and at the 90th percentile it is less than 100. This indicates that there is little or no programming activity taking place on more than half of the projects. Examining the number of message authors across all forums shows that only projects at the 90th percentile and above have more than two contributing message writers.

To better grasp the structure of the data, it is convenient to represent it graphically. Phenomena with highly skewed distributions often follow "power-law distributions" — roughly, the relationship between the frequency of an event and its size appears linear on a log-log scale. Zipf's law, a subspecies of power-laws, says that the size (y) of the r'th largest occurrence of an event will be inversely proportional to its rank-order: $y \approx r^{-b}$, with $b \approx 1$ (Zipf 1949).[7] Distributions of this sort are very common across a wide range of natural phenomena — the frequency of earthquakes is a stock example, with small quakes being very common compared to the few large ones. For social phenomena, power laws have been observed for city size (Krugman 1996), formal organizations such as the military and hospitals (Mayhew and Rushing 1973; Mayhew and James 1972), and — relevant in this context — aspects of the flow of Internet traffic (Huberman 2001).[8] We find that the OSS community displays a very similar structure. The panels in Figure 2 show rank-ordered distributions for six activity measures. In each case, projects are rank-ordered logarithmically on the x-axis, ordered from highest to lowest. Thus, for the 'Developers' panel, the first tick on the x-axis marks the project at the 99.995th percentile, the second the project at the 99.99th percentile, and so on. Given that each plot shows the data for more than 45,000 projects (rather than, say, a fitted line), the regularity is striking. In each case, the activity is overwhelmingly concentrated in the very upper end of the distribution. That this should be so for the number of downloads is consistent with other work the topology of the Internet (Faloutsos, Faloutsos, and Faloutsos 1999). But it also holds for the measures which more closely index actual interaction, such as the number of developers and the number of messages. Mayhew et al. 1995 found similar patterns in naturally occurring

[7]Power-law distributions express a relationship between size and frequency; Zipf's law relates rank and frequency. The two are closely related. See Adamic for a helpful discussion of relationships of this type.

[8]There is an under-explored link between the work of Mayhew and his associates and more recent studies of the emergent structure of Internet communities (and other social phenomena with power-law-type characteristics). Mayhew's work is strongly oriented towards the deep problem of emergent differentiation within social aggregates (Blau 1977, 1970), and thus might well provide a fruitful sociological approach to these issues. Pursuing this link in more detail is a task for future research.
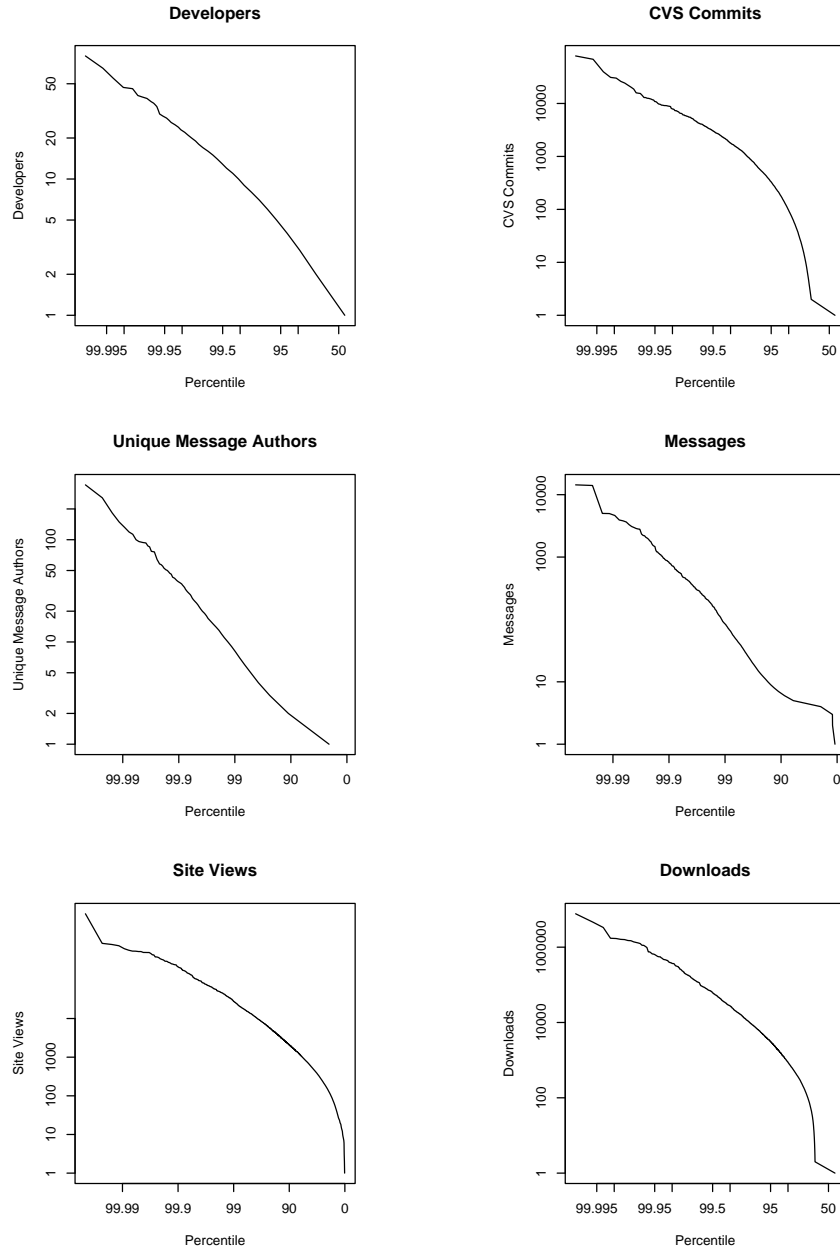
face-to-face groups.

Figure 2: *Six measures of project activity.*

It is interesting in and of itself that each of these measures of oss community activity should be so consistent. Yet this is only a first step towards understanding the development of the field. The mechanisms generating these consistent rank distributions may be quite different from one another. Different kinds of activities may cluster in different sorts of projects, for instance, even though the shape of the overall distributions is the same. To illustrate this point, we compared the most active projects as measured by number of downloads (Table 2) with the most active projects as measured by number of cvs commits (Table 3). The most downloaded projects are mainly end-user applications. The most heavily developed projects are mainly "behind the scenes" system-level applications, programming environments, or utilities providing basic functionality to an Operating System.

In summary, we found power-law type distributions for all activity measures in our dataset. In each case, a tiny number of projects dominate an activity-type when measured by volume. Different projects (and different kinds of project) dominate different measures. Measures of user interest in a project — such as site views and downloads — are not closely related to measures of developer activity on a project. The oss community presents different faces to different audiences. Most users will download a similar set of applications (the most popular ones), but most project developers will never see a user looking to download their software. Many users may contribute bug reports or feature requests, but the vast majority of projects do not receive them.

DISCUSSION

These preliminary findings raise questions about the scope and accuracy of existing theories of oss development. In particular, the highly stratified structure of project development suggests that broad generalizations about the "oss approach" or the "oss community" or "oss developers" may be much too broad. Similarly, arguments about what makes oss successful may need tempering given data on the characteristics of a typical project. So, for instance, Raymond's image of the bazaar does not capture the fact that the typical project has one developer, no discussion or bug reports, and is not downloaded by anyone. "Linus's Law," coined by Eric Raymond, states that "Given enough eyeballs, all bugs are shallow."[9] It is hard to see how it can apply to most oss projects in our dataset, as the only eyeballs looking at the code belong to the lone developer. Similarly, the transactions costs model presented by Benkler argues that the flow of information in a pervasively networked environment tends to allocate the best human capital to the job at hand. There is little evidence of this process at work in the

---

[9] An amazing three metaphors packed into seven words.

Table 2: Top Ten Most Downloaded Projects

Name	Type of Application
CDex	Audio
VirtualDub	Video processing
ZSNES	Game emulator)
Back Orifice 2000	System/network admin
MySQL	Database
Dev-C++	Development
AFPL Ghostscript	Print/typesetting
MyNapster	File sharing
Tux Racer	Game
Gnucleus	File sharing

present sample of OSS projects, if only because there simply is so little communication registered on the vast majority of projects.[10]

What can we make of this gap between theory and data? A natural response is to argue that what is at issue here is the appropriate definition of the OSS community or the OSS field of projects. On this view, many of the projects hosted by Sourceforge have fallen dead-born from the PC. They are unlikely to develop beyond a vague plan for development, a general invitation to help out, and a version 0.001a alpha release (if even that). So perhaps we should exclude them from the pool of projects we consider when we talk about OSS. An argument in favor of this view is that the Sourceforge data has a disproportionate number of projects that are likely to fail or have already failed. If we look instead to projects that have succeeded — the standard cases studies like Apache, or the Linux Kernel and so on — then we see the true OSS model at work.

A difficulty with this argument is that it simply restates the problem by redefining the scope of the term "OSS community." In addition, it contradicts the rhetoric that evangelists for OSS use and theorists echo. Why shouldn't the thousands of amateur developers count as part of the community? Isn't that what is supposed to characterize the phenomenon in the first place?

A better approach, we argue, is to see the highly stratified nature of OSS as an opportunity for advancing theory. It seems clear that for every successful OSS project there are thousands of unsuccessful ones. This surely raises important questions about the alleged benefits of the approach. It obviously does not mean OSS cannot work (any more than the failure of regular businesses means that capitalism cannot work), but it

---

[10] We do not discuss a third strong claim often made for the OSS model, namely that it produces innovative software better than standard approaches. There is not a great deal of evidence for this claim in our data, but this is a complex issue that needs separate treatment at greater length.

does mean that the scope conditions for a successful project need to be better specified. Why, for example, does Apache succeed when so many others fail? It cannot simply be because it's an OSS project, for so are the failures. There must be something else at work.

What might that be? We do not pretend to offer a full explanation here. But we do want to suggest some possible approaches, grounded in sociological approaches to organization and mobilization. Three aspects of OSS development seem particularly under-theorized. First, it is clear from the case-study literature that successful OSS projects are most often staffed (at their cores) by professional software developers. This might seem like a banal observation, but we should bear in mind how this differs from the populist image of teenage hackers in their parents' basements beavering away on new technologies. Despite the public image of hackers, the projects that matter are (more often than not) run by professionals and not amateurs. We need more research into how this group regulates itself, and what its values are. A hypothesis for future research is that *the more successful an OSS project, the more professional its core contributors will be*, as measured by length of practical experience, formal qualifications or both.

Second, we suggest that the data presented here, corroborated again by case studies, implies that the role of project leaders in mobilizing development is crucial. Effective project leadership seems to us one of the most likely candidates for differentiating successful projects from unsuccessful ones. The voluntary nature of participation in an OSS project makes the role of the leader vitally important. The preconditions for successful mobilization may well be best understood not via economic approaches, we argue, but via concepts borrowed from the literature on social movement organization and political action. The theoretical literature on OSS at present is dominated by ideas derived from Economics, Law and Management. These disciplines obviously have important contributions to make, but may tend to downplay or miss key aspects of the OSS phenomenon. As we said at the beginning, OSS is a hybrid — part economic project, part network organization and part social movement. The two latter aspects, and especially the last, seem the least well-understood. We suggest that *successful OSS projects will tend to have core participants mobilized in a way similar to core participants in successful social movement organizations.*

Third, and last, we argue that the importance of hierarchical organization to successful OSS projects is systematically underplayed in the theoretical literature. We suggest that *successful OSS projects will tend to have a strong hierarchical component*, at least in the ways they manage the relations between lead (and core) developers and other contributors. This is not an original observation on our part, yet it is remarkably absent from most of the standard accounts of OSS, which tend instead to focus on the (allegedly) quasi-anarchic qualities of the development process. Jordan Hubbard, a leading contributor to the FreeBSD project, comments:

Table 3: Top Ten Projects by CVS Commits

Name	Type of Application
Crystal Space 3D Engine	Graphics engine
Open CASCADE Auto Config	Development
Direct Rendering Infrastructure	Graphics development
Squid HTTP Proxy Developments	Proxy server
gkernel	Core OS development
Linux Standard Base	Core OS development
LinuxSH	Core OS development
QuakeForge	Game engine
phpGroupWare	Web groupware
Python	Programming language

Despite what some free-software advocates may erroneously claim from time to time, centralized development models like the FreeBSD Project's are hardly obsolete or ineffective in the world of free software. A careful examination of the success of reputedly anarchistic or "bazaar" development models often reveals some fairly significant degrees of centralization that are still very much a part of their development process.[11]

There is a lot to be said for this argument. As several case-studies have noted, core development on successful OSS projects tends to be well organized. In some cases — most conspicuously in the case of the Linux kernel — it is entirely hierarchical, with lead developer Linus Torvalds deciding which patches will be accepted to new versions of the kernel and which will not.[12] We suggest that, despite the canonical image of OSS development as a free-for-all bazaar, hierarchical organization is central to the success of important projects. This hierarchy is not a formal organizational chart but rather (we conjecture) a status-based pecking order which is known to project participants and serves as a way of policing members. Following Arthur Stinchcombe's observation that contracts may have hierarchical elements (Stinchcombe 1985), we suggest that the apparently open-form, flat networks of the OSS community are in many ways strongly hierarchical. We further suggest that, contrary to the conventional wisdom, hierarchy tends to emerge as a precondition of successful project management and hypothesize that *the closer a successful project is to the core of the OSS community, the more hierarchy will be found in its management style*. Thus, for instance, the social organization of

---

[11]Quoted in *Linux Magazine*, April 2000. http://www.linux-mag.com/2000-04/opensource_evol_01.html.

[12]This has created a significant amount of tension in the kernel hacker community. [REF]

kernel hackers will be more hierarchical than that of developers of add-on applications for the GNOME or KDE desktop environments, because the kernel is the essence of the operating system, whereas additional text editors or desktop calculators are much less important.

In sum, we have argued that the huge gap between successful and unsuccessful (active and inactive) projects in our data is a real puzzle. We offer these hypotheses as a way of focusing attention on some aspects of the OSS community which we feel have been neglected in current theory. We have consciously formulated them in a way that focuses on aspects of OSS organization — professionalism, clear leadership, hierarchy — that are antithetical to the standard image of the community. More generally, we have argued that researchers should attend more closely to the social structure of the OSS community. The process of OSS development is embedded in particular structural and organizational contexts that theorists of OSS have so far paid little attention to. Investigating them offers a promising route for an original sociological perspective on this exciting phenomenon.

REFERENCES

Abbate, Janet. 2000. *Inventing the Internet.* Cambridge, MA: MIT Press.

Adamic, Lada A. *Zipf, Power-laws, and Pareto — a ranking tutorial.* Internet Ecologies Area, Xerox Palo Alto Research Center, Palo Alto CA. http://ginger.hpl.hp.com/shl/papers/ranking/ranking.html.

Benkler, Yochai. 2002. "Coase's Penguin, or, Linux and the Nature of the Firm." *Yale Law Journal*, Forthcoming, Winter.

Blau, Peter M. 1970. "A Formal Theory of Differentiation in Organization." *American Sociological Review* 35:201–18.

———. 1977. *Inequality and Heterogeneity: A primitive theory of social structure.* Glencoe, IL: Free Press.

Chandler, Alfred D., and James W. Cortada, eds. 2000. *A Nation Transformed by Information.* Cambridge, MA: Harvard University Press.

Coase, R.H. 1988. *The Firm, the Market and the Law.* Chicago: University of Chicago Press.

Dertouzos, Michael. 1997. *What Will Be: How the New World of Information Will Change Our Lives.* New York: Harper Business.

DiBona, Chris, Sam Ockman, and Mark Stone, eds. 1999. *Open Sources: Voices from the Open Source Revolution.* New York: O'Reilly.

DiMaggio, Paul, Eszter Hargittai, W. Russell Neuman, and John P. Robinson. 2001. "Social Implications of the Internet." *Annual Review of Sociology* 27:307–336.

Faloutsos, Michalis, Petros Faloutsos, and Christos Faloutsos. 1999. "On power-law relationships of the Internet topology." In *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication,* 251–262. Cambridge, Massachusetts, United States: ACM Press. doi:http://doi.acm.org/10.1145/316188.316229.

Gilder, George. 1990. *Microcosm: The Quantum Revolution in Economics and Technology.* New York: Simon / Schuster.

Harrell, Frank E. 2001. *Regression Modeling Strategies.* New York: Springer.

Hippel, Eric von. n.d. "Open Source Shows the Way - Innovation By and For Users - No Manufacturer Required." http://opensource.mit.edu/papers/evhippel-osuserinnovation.pdf.

Huberman, Bernardo. 2001. *The Laws of the Web.* Cambridge, MA: MIT Press.

Ihaka, Ross, and Robert Gentleman. 1996. "R: A Language for Data Analysis and Graphics." *Journal of Computational and Graphical Statistics* 5 (3): 299–314.

Krishnamurthy, Sandeep. 2002. "Cave or Community? An Empirical Examination of 100 Mature Open Source Projects." *First Monday* 7 (6). %5Curl%7Bhttp://firstmonday.org/issues/issue7_6/krishnamurthy/index.html%7D.

Krugman, Paul. 1996. *The Self-Organizing Economy.* New York: Blackwell.

Lakhani, Kareem, and Eric von Hippel. n.d. "How Open Source Software Works: Free User to User Assistance." *Research Policy*, Forthcoming.

Lancashire, David. 2002. *Code, Culture and Cash: The Fading Altruism of Open Source Development.* http://www.firstmonday.org/issues/issue6_12/lancashire. http://www.firstmonday.org/issues/issue6_12/lancashire.

Lerner, Josh, and Jean Tirole. 2000. "The Simple Economics of Open Source." National Bureau of Economic Research (NBER) Working Paper 7600.

———. 2002. "Some Simple Economics of Open Source." *The Journal of Industrial Economics* L:197–234.

Lessig, Lawrence. 2001. *The Future of Ideas: The Fate of the Commons in a Connected World.* New York: Random House.

Mayhew, Bruce H., and Thomas F. James. 1972. "System Size and Structural Differentiation in Military Organizations: Testing a Harmonic Series Model of the Division of Labor." *American Journal of Sociology* 77:750–765.

Mayhew, Bruce H., J. Miller McPherson, Thomas Rotolo, and Lynn Smith-Lovin. 1995. "Sex and Race Homogeneity in Naturally Occurring Groups." *Social Forces* 74:15–52.

Mayhew, Bruce H., and William A. Rushing. 1973. "Occupational Structure of Community General Hospitals: The Harmonic Series Model." *Social Forces* 51:455–462.

Mockus, Audris, Roy T. Fielding, and James Herbsleb. 2000. "A case study of open source software development: the Apache server." In *Proceedings of the 22nd international conference on Software engineering,* 263–272. Limerick, Ireland: ACM Press. doi:`http://doi.acm.org/10.1145/337180.337209`.

Moody, Glyn. 2001. *Rebel Code: Linux and the Open Source Revolution.* New York: Perseus.

Neff, Gena, and David Stark. 2002. "Permanently Beta: Responsive Organization in the Internet Era." Center on Organizational Innovation Working Paper, Columbia University, September.

Perens, Bruce. 1999. "The Open Source Definition." In *Open Sources: Voices from the Open Source Revolution,* edited by Chris DiBona, Sam Ockman, and Mark Stone, 171–188. Sebastopol, CA: O'Reilly.

Raymond, Eric S. 1998. *The Cathedral and the Bazaar.* `http://www.firstmonday.dk/issues/issue3_3/raymond/`.

———. 2001. *The Cathedral and the Bazaar.* New York: O'Reilly.

Sandred, Jan. 2001. *Managing Open Source Projects.* Wiley.

Schweik, Charles M., and Andrei Semenov. 2003. *The Institutional Design of Open Source Programming: Implications for addressing complex public policy and management problems.* `http://www.firstmonday.org/issues/issue8_1/schweik/index.html`.

Stinchcombe, Arthur. 1985. "Contracts as Hierarchical Documents." In *Organization Theory and Management,* edited by Arthur Stinchcombe and Carol Heimer, 121–71. Oslo: Norwegian University Press.

Sunstein, Cass. 2001. *Republic.com.* Princeton, NJ: Princeton University Press.

Vixie, Paul. 1999. "Software Engineering." In *Open Sources: Voices from the Open Source Revolution,* edited by Chris DiBona, Sam Ockman, and Mark Stone, 91–100. Sebastopol, CA: O'Reilly.

Wershler-Henry, Darren. 2002. *Free as in Speech and Beer: Open Source, Peer-to-Peer and the Economics of the Online Revolution.* New York: Prentice Hall.

Williams, Sam. 2002. *Free as in Freedom: Richard Stallman's Crusade for Free Software.* Sebastopol, CA: O'Reilly.

Zipf, G.K. 1949. *Human Behavior and the Principle of Least Effort.* Reading, MA: Addison-Wesley.