

# The Evolution of Governance Conventions in Open-Source Projects

Suresh Naidu\*

February 22, 2004

## Abstract

I focus on the emergence of governance norms within Open-Source software projects, arguing that they are neither decentralized nor authoritarian, but instead what some anthropologists call "reverse-dominance" hierarchies. I also show that Eric Raymond's assertions about the norms of property ownership as a primary governance mechanism are to be qualified by noting the overwhelming prepondance of deliberative decision-making. In illustrating this, I extend and analytically solve a multi-level evolutionary model of conflict-resolution norms, initially proposed by Maynard-Smith(1982) and extended and simulated by Bowles and Choi(2003), using recent techniques in stochastic game theory developed by Young(1993), Kandori, Mailath, Rob(1993), and Ellison(2000).

---

\*University of Massachusetts-Amherst and the Santa Fe Institute. snaidu@santafe.edu. Incredibly preliminary, so please don't cite without permission. Thanks are due to Alessandra Rossi and Jung-Kyoo Choi for comments and the Santa Fe Behavioral Science Discussion Group for tolerating a presentation of the ideas in here. All errors, and there almost certainly are errors, are mine.

New economical and social institutions, in so far as they were a creation of the masses, new ethical systems, and new religions, all have originated from the same source, and the ethical progress of our race, viewed in its broad lines, appears as a gradual extension of the mutual-aid principles from the tribe to always larger and larger agglomerations, so as to finally embrace one day the whole of mankind, without respect to its diverse creeds, languages, and races. – Peter Kropotkin, "Mutual Aid".

## 1 Introduction

The shift towards information technology that has marked the New Economy is precipitating extraordinary institutional changes, most of which are still in the process of unfolding. One of these changes can be found in the surprising success of Open-Source Software (OSS), a very large online community of volunteer developers and users that provide high-quality software to the public at no charge. Developing countries, such as Brazil and South Korea, and large corporations, such as IBM, are installing exclusively Open-Source software on billions of dollars worth of computers, as well as actively supporting the Open-Source process. The Open-Source community has transformed itself from a marginal network of technophiles to a mainstream alternative to commercial software, and has already shown great potential to alter the way many knowledge goods are produced and disseminated.

## 2 Open-Source Institutions

The last decade has witnessed the not-slow growth of a mid-sized army of volunteer software developers. What is most novel is not their motivation to write software; the challenge and craft of developing good programs came hand-in-hand with the computer. What was new was the large-scale, relatively decentralized, functional organization of software developers enabled by the combination of new property right regimes, modes of coordination, and motivations to help users and developers alike.

Open-Source software offers a wealth of institutional innovations for study, and constitutes an area ripe for social scientists. Economists, however, despite some notable exceptions(Lerner and Tirole 2000, ?, ?) have been largely silent on analyzing the mechanisms, rules and norms that facilitate this type of non-market, large-scale decentralized coordination. However, other social scientists, especially lawyers(?, Benkler 2003, McGowan 2001) have been working on explaining and understanding the history and mechanisms of OSS development.

A particularly outstanding innovation in the Open-Source community is the peculiar license structure, most famously incorporated in the GPL(GNU public licence), which demands that any developer, volunteer or commercial, that modifies software released under the license must also freely distribute the modified source code as well. This has been called "copyleft" by some (Benkler 2003), and is presumably enforceable in court, although it has not faced the acid test of the judicial system as of yet. Many pages have been written in the past few years puzzling over the legal and economic significance of the licences, and they remain a phenomenon of substantial interest, particularly to economists interested in property rights and institutions. Siobhan O'Mahony (OMahony 2003) documents the various mechanisms, formal and informal, open-source communities have in place to protect their collective property from would-be monopolists.

A fair amount of intellectual energy has been expended on the motivations behind OSS development. Many social scientists(Lerner and Tirole 2000) have been very consistent in their application of rational choice to explain the motivations underlying OSS participation. Others have been less parsimonious, invoking a diverse set of motivations, including status, reputation, altruism, reciprocity, identity. Yochai(Benkler 2003) has even put forward the possibility that individual motivations are not the explanandum of primary interest in OSS, but rather the institutions that coordinate this large volume of contributions. People have expended their leisure time in innumerable ways; software development is but one of them.

I am sympathetic to Benkler's view, and so this paper will take a different tack. Rather than focusing on individual motivations for participation, I will look at OSS as a unique opportunity to study the emergence and complexity of community governance, and propose it as an instance (prominent, but not unique), of team production without a coercive apparatus. In doing so, perhaps some light can be shed on similar institutional architectures, ranging from foraging hunter-gatherer bands to other volunteer organizations. Many social institutions share the ability to organize production without a monopoly on the right to sanction, be it the application of the state stick or the withdrawal of the wage carrot.

### 3 Inverted Hierarchies for GNU and Gazelle

The allocation of decision-making power in OSS projects is a matter of some controversy. At first glance, OSS projects are archetypical pyramids, with the project owner(s) at the top, with the ability to unilaterally veto proposed changes to the source code tree. Why have such authoritarian governance structures emerged in an atmosphere of voluntary labor supply? Many authors (Rossi 2004, McGowan 2001, Healy and Schussman 2004, Lerner and Tirole 2000) have noted the puzzle of volunteer labor working in formally autocratic structures. The *de jure* relationships are indeed distinctly asymmetric, with formal rights to veto contributions generally monopolized by a single individual or collective of individuals. Linus Torvalds, the founder of Linux, retains the right to exclude any piece of contributed code, or introduce any of his own. In some cases these rights are held by a committee or collective, such as the Apache web server. Lerner and Tirole note that "in a number of instances, such as Linux, there is an undisputed leader. While certain aspects are delegated to others, a strong centralization of authority characterizes these projects" (pg 22). McGowan implicitly assumes that OSS projects are hierarchies in the sense that firms are hierarchies, and then uses reputation, asset specificity, and organizational loyalty to explain why developers tolerate these totalitarian political arrangements.

However, this *de jure* authoritarianism may mask substantial *de facto* equality. What the authors above fail to recognize is the limited scope and substantial constraints faced by the formal owners/managers of the projects. Tirole and Lerner in fact explicitly take the opposite view, arguing that there is a lack of formal authority and a substantial amount of real authority. By the definition of authority they use, from Aghion and Tirole(?), it is puzzling precisely how the project leader can, either formally or informally, choose the action of any of the developers. They justify this by noting that the project leader's recommendations tend to be followed by the mass of developers, and that the project leader cajoles other agents out of "forking" (more about this later). However, they immediately note that a key to good leadership is trust, and that leaders' motives must be seen to not "polluted by ego-driven, commercial, or political biases". One wonders, then, exactly what the scope of authoritarian behavior on the part of leaders is? I suggest it is minimal, and that leadership, while serving an important coordinating function, is unattached to any real authority, in the sense of wielding unilateral sanctions.

Project leaders implement collective standards of quality, and their decisions are tacitly ratified by the continuing participation of the development community. While leaders can revise the codebase unilaterally, they cannot use this ability to violate technical standards or community norms. While the owners/leaders do provide an important coordinating and leadership functions, they cannot exercise unilateral decision-making power and maintain project viability. They also cannot dictate project direction, in the sense of demanding certain tasks of the community. The project manager retains no ability to sanction, either positively or negatively, in an open-source project. If we use the term power to mean the ability to coerce others, then no single agent, not even the project manager, has any power. Their role is more like the conductor of a volunteer orchestra, or the coach of an intramural team, than a boss in the firm.

An insight from anthropology may be useful in clearing up this puzzle. Eric Raymond (Raymond 1999) famously made the analogy between early hunter-gatherer tribes and OSS projects, albeit with a distinct lack of data or formal analysis. Using more rigorous anthropological and sociological evidence, formally modelling the process of norm formation, and drawing on less introspective data should allow one to extend

and evaluate the claims made by Raymond. Mining anthropology for insights into contemporary human institutions is not new, nor is it particularly controversial, but the analogies should be made formal, in order to elucidate precisely what the common mechanisms are.

A phenomenon that has fascinated anthropologists for decades is the stable egalitarianism of foraging cultures. Chris Boehm, in "Hierarchy in the Forest" (Boehm 1999), has coined the term "reverse-dominance hierarchy" to describe the political arrangements that sustain such egalitarianism. Boehm does not deny the large inequality in ability among hunter-gatherers, nor the existence of leadership or coordination-enhancing hierarchies, but he identifies how collective sanctions, from ridicule to assassination, prevent people from translating a productivity advantage or coordinating role into a political or wealth advantage. In brief, Boehm's thesis is that the lower-ranked members of a tribe band together to sanction people in leadership positions, very talented hunters, and other would-be tyrants. The hierarchy still exists, but the flow of power is reversed; the many at the bottom dominate the few at the top. Although it is not emphasized in this paper (see Appendix) I would also argue that the inequality of productivity is a corequisite for stable political egalitarianism; one needs a hierarchy of talent in order to effectively reverse the flow of power.

I use the term political here in a wide sense; politics is about the allocation of power, and power is defined in the standard way (Dahl 1957, ?): "For B to have power over A, it is sufficient that by imposing or threatening to impose sanctions on A, B is capable of affecting A's actions in ways that advance B's interests, while A lacks this capacity with respect to B". By this definition, I argue that project leaders have little power over individual developers, but the collective of participating developers retain a degree of power over the project leadership.

This political structure is maintained crucially by a strong value placed on autonomy. Foraging males place a premium of political self-determination, and are very reluctant to cede control to anyone. Similarly, Eric Raymond, in a recent interview, remarked of Linux developer culture that "Some obvious characteristics are distrust of control and hierarchy.". A peek at the e-mail archives of many open-source projects will reveal the fierce bickering that occasionally erupts over project direction. There is no uncontested downward cascade of directives from project leaders to developers. Continuing the analogy with foraging cultures, Boehm points out: "In societies that are egalitarian, subordinate intransigence is pervasive and results in vigilant suppression of alpha-type behaviors" (Boehm, pg 169).

Boehm is keen on explaining why such features evolved among humans, and not other primates. A key portion of his explanation is that the development of weapons was a great equalizer, in that the probability of a strong despot being killed becomes substantially larger. Hillard Kaplan and Jane Lancaster (Kaplan and Lancaster 1984) have noted that the credible threat of departure is an even stronger impetus to egalitarian relationships in foraging projects. With high mobility comes egalitarianism, a thesis that accounts for the emergence of the first states in environments with a steep resource gradient, like the banks of the Nile and Euphrates rivers, where a dominated agent cannot relocate to a productive patch of land easily. The similarity to an environment of volunteer labor should not be missed, particularly when the productive assets are non-rival, so any agent can walk off with the assets they need to produce (e.g. no scarcity of land or game).

The political mechanisms that stabilize the governance conventions of Open-Source projects are also isomorphic to those that maintain the egalitarian norms of foraging projects. There is indeed substantial asymmetry among the relative contributions of programmers in Open-Source projects. However, de facto decision making power is distributed much more equally than either strict valuation of contribution or formal allocation of decision-making power would indicate. This is because the non-alpha programmers, those minor patch contributors and debuggers, documenters, and other providers of essential components of OSS development collectively retain the ability to leave the project, or engage in very public verbal sanctions, should the project elite, be it an individual or a project, attempt to exercise too much power (e.g. dictate terms to developers). The analogue to collective sanctions or exodus in Open-Source communities is forking the project. A project is said to fork when a subproject of agents, for whatever reason, copy the source tree and start a rival project. This is generally considered to be a very bad thing for the project, as it splits the talent pool working on the software.

The implication of this is that while project organization may appear formally quite authoritarian, a vibrant egalitarianism emerges when real decisions have to be made. As Boehm notes "A foraging band may have a formal leader who is referred to as the headman or some such title, or it may have an informal leader who steps forward to help in decision-making as long as the band welcomes him in doing so.[...] In every case, such a person is respected, and the danger exists that he (or she) may try to take advantage of this position and selfishly parlay it into something more." A similar architecture of authority operates within the acephalous communities of OSS, recognizing alpha-programmers, but not conceding any substantial coercive power to them. Even Larry Wall, alpha-hacker behind Perl, cannot command OSS developers to work on any particular piece of code.

Consider the organization of Linux. Linus Torvalds cannot unilaterally undertake decisions without securing the tacit consent of a sizeable portion of the Linux community. This explains why developers voluntarily participate in hierarchical organization; the hierarchy empowers the people at the bottom, rather than the people at the top. There are numerous examples of this political threat of forking constraining decisions even in the most hierarchical OSS projects (Moody 1991). The collective threat of punishment makes sure that leaders make decisions in line with the technical standards and community of the project, does not violate anyone's autonomy, and cannot enrich themselves at the expense of the rest of the project.

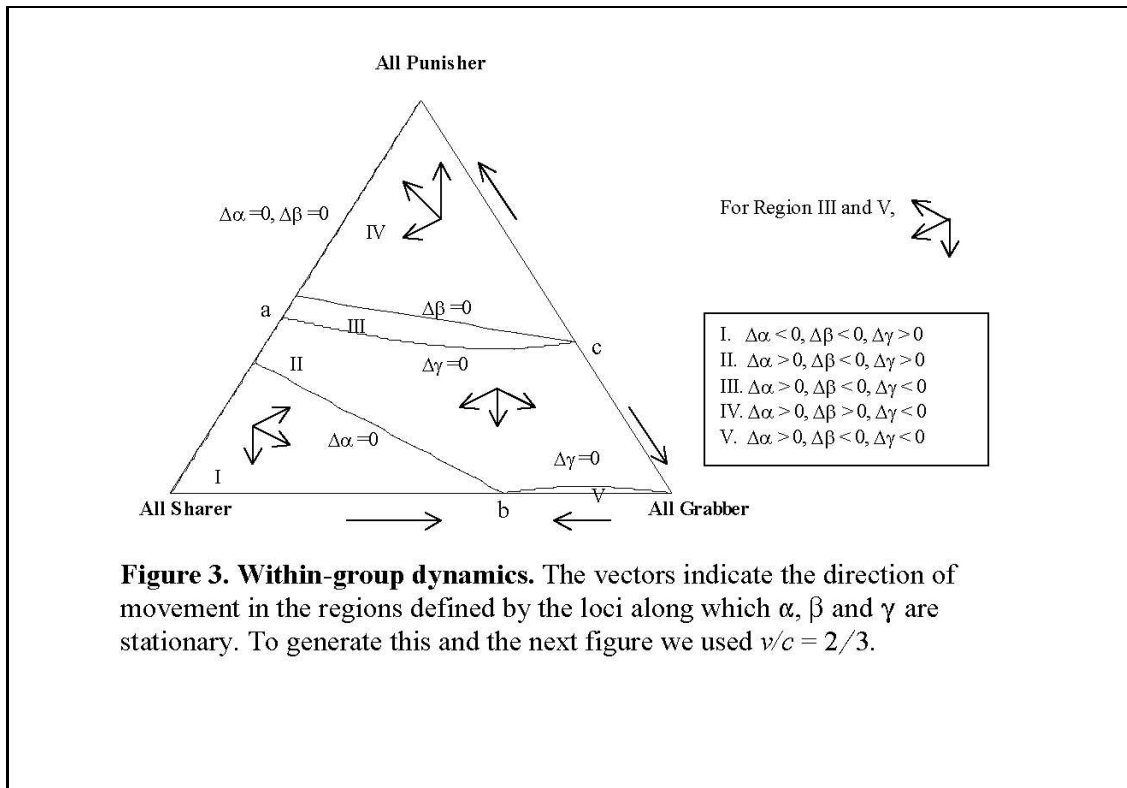
While this might appear as a minor point, it is a substantial challenge to the theory of the firm proposed by Alchian and Demsetz (Alchian and Demsetz 1972), where team effort is costly to monitor, and therefore the monitor should be given residual claimancy over the output and the right to terminate employees. OSS projects manage to solve the team effort supply problem without allocating residual termination rights to any one agent. Agents can be coordinated and high-quality can be elicited, with the need for sanctioning ability. No single person has coercive capabilities in an OSS project, yet a large number of diverse contributions manage to get coordinated and integrated. The Hobbesian organization of team production gives way to a Rousseauian one, but how much of this is due to the peculiar motivations around effort supply is still an open question.

Of course, this analogy cannot be taken too far, and must be disciplined. The theoretical model below illustrates the mechanism that generates equality is shared by foraging projects and open-source software, abstracting from the countless differences that exist between foraging bands of the late pleistocene and contemporary tribes that conduct most of their interaction via e-mail.

## 4 The Evolution of OSS Governance Conventions

This model is a stochastic evolutionary variant of the simulation conducted in Bowles and Choi (Bowles 2004). I add two things: agent heterogeneity (in appendix) and an analytical solution using recent techniques in stochastic game theory. Without diminishing the power of agent-based simulations, pushing analytical results as far as they can go is a helpful disciplining device. Simulations are often useful, but they are often less transparent and general than the corresponding theoretical models. I therefore model conflict, project-selection and conformity as a perturbed Markov process, following Young (Young 1998). I also make use of analytical results by Ellison (Ellison 2000).

First, consider an isolated project. We have three possible strategies, Dominators, Contributors, and Autonomists. This are analogous to the Grabber, Sharer, Punisher strategies in Bowles and Choi. Agents meet in conflicts. Dominators attempt to win every conflict, and fork a project if they lose a conflict with another Dominator. Contributors always compromise in conflicts with each other and Autonomists, and surrender to Dominators. Autonomists compromise with each other and Contributors, and collectively fork the project should they enter into a conflict with a Dominator. The payoffs are given by the following payoff matrix. Note that  $N$  is the project size,  $\beta$  is the fraction of the population that are Autonomists, and  $C > V$ :



**Figure 3. Within-group dynamics.** The vectors indicate the direction of movement in the regions defined by the loci along which  $\alpha$ ,  $\beta$  and  $\gamma$  are stationary. To generate this and the next figure we used  $v/c = 2/3$ .

strategy	D	C	A
D	$V - C/2$	$V$	$(1 - \beta)V - \beta C$
C	0	$V/2$	$V/2$
A	$\beta V - (1 - \beta)C$	$V/2$	$V/2$

The game dynamic is mixed best-response, where agents choose a distribution of strategies in response to the average payoffs in the previous period. Formally, then, our state space is  $X = (b, a)$ , where  $b$  is the number of Autonomists and  $a$  is the number of Contributors. Consider the following simplex diagram from Bowles and Choi(2003), which summarizes the unperturbed Markov process.

Let  $V$  and  $C$ ,  $C > V$  be such that  $n^* = V/C$  is a Nash equilibrium, which Bowles and Choi dub Hobbesian due to the preponderance of conflict, and let  $m^* = \bar{a}$  be the number of Contributors that must coexist with Autonomists in order to make an agent indifferent between Dominator and Autonomist A simple calculation shows us that  $m^* = N(1 - \sqrt{\frac{V}{2(V+C)}})$ . The set  $R = \{(N - m, m) | m \geq m^*\}$  as another set of Nash equilibria, which Bowles and Choi call Rousseauian. I consider only symmetric Nash equilibria, on the grounds that the population will always choose the same strategy distribution in the unperturbed Markov process. Since our dynamic is a best-response, these are also the absorbing states of the unperturbed Markov process, abstracting from integer problems.

Following Young(Young 1998) and Kandori-Mailath-Rob(Kandori, Mailath, and Rob 1993), I add a small mutation term. In this case, each agent, with probability  $\epsilon$  plays a strategy drawn at random from the three pure strategies(i.e. with a uniform distribution). Define the cost (Kandori, Mailath, and Rob 1993), or resistance(Young 1998) of a transition as  $c(n|m) = \lim_{\epsilon \rightarrow 0} \log(P^\epsilon(n|m))/\log(\epsilon)$ . This is the number of mutations required to transition from  $m$  to  $n$ . We know that if the mutation rate is small, the system will spend most of its time in the absorbing states(or recurrent classes) of the unperturbed process, and the question is which one of the absorbing states will it spend most of its time at. This is called the stochastically stable state, and is identified by being that state which is the easiest(in terms of least number of mutations) to get to to

any other. That is, if the only two absorbing states(or recurrent classes) are A and B, A is stochastically stable if it is costlier to transition from A to B than the transition from B to A.

The rest of the model will be characterizing the stochastically stable state, i.e. that state we expect to obtain after a sufficiently long period of time. The choice of equilibrium concept is not arbitrary: stochastic stability has three desirable properties that distinguish it other equilibrium concepts. Firstly, it is the natural analytical prologue to many agent-based simulations, based as it is on Markov processes with small perturbations. An agent-based model is essentially a Markov process with a very large state space, with random elements introduced to guarantee a unique long-run outcome. Secondly, it has the very realistic property that it does not require stationarity of the agent's actions to characterize the average state of the system. The resulting equilibrium is understood to be a statistical phenomena, and not necessarily a property of the system at any given time. Thirdly, if we assume that many social norms are best understood as robust Nash-equilibria, then explicitly introducing random shocks is important for determining which norms will prevail over the long-run. People make mistakes, innovate, and experiment with conventions all the time; explicitly introducing such phenomena are essential for understanding why certain conventions might be more robust or persistent.

**Proposition 1**  *$n^*$  is the stochastically stable state*

Proof: By Ellison(Elison 2000), suffices to show that  $R^*(n^*) > CR^*(n^*)$ , where  $R$  is the radius (the cost of leaving the state), and  $CR^*$  is the modified coradius(the cost of returning to the state). The number of mutations needed to transition into the basin of attraction of  $R$  is  $\frac{V^*N}{2(V+C)}$ , but the number of mutations required to get from  $R$  into the basin of attraction of  $n^*$  is at most  $N - m^*$ , although there exists a path of length  $m^* + 1$  along the side of the simplex. If we were stuck here, the situation would be indeterminate, depending on the relative size of  $m^*$ . However, due to Ellison's result, we can do better than this, by using the path of absorbing states in  $M$  to  $n^*$  as the path for transitioning to the Hobbesian equilibrium. It is thus possible to use the modified Coradius  $CR^*(n^*) = m^* + 1 - m = 1^*$ , which is 1, since we subtract the radii of all the intermediate absorbing states between a state in  $M$  and  $n^*$ . The result follows immediately since for sufficiently large  $N$ , we have  $V^*N/2(V+C) > 1$

This happens to be a model in which Ellison's insight, that a sequence of short steps occurs much more often than a single large jump between absorbing states, applies exactly. Drift can quickly unravel a Nash equilibrium. However, the model as it stands is not a very convincing cartoon of either my argument or a real Open-Source project. A few additional features are needed.

## 4.1 Project Competition

One interesting feature of the open-source world is the competition for volunteer developer talent. Projects are operating in an environment where labor is the scarce resource. Projects that can deliver higher developer satisfaction will attract more skilled volunteers. A component of this is certainly interesting technology and talented peers, but also, given the preferences anecdotally mentioned by Raymond, it seems that a key ingredient is delivering good governance. Projects need to be seen as good environments to work in, with plenty of skill-sharing, fascinating technology, and no shortage of prosocial behavior.

Projects are trying to attract developers, thus project leaders need to exhibit the characteristics of accountable coordinators willing to listen to the other contributors. The ruthlessness of the market for good projects is a feature neglected in much of the literature. Projects must show themselves to be good work environments in order to attract developers. Projects with good governance will thus attract developers from poorly-governed projects, and then spin-off their own new projects. An interesting irony is that competition breeds egalitarianism, because volunteer labor is the scarce input. Labor really does hire capital(as it were) in the Open-Source bazaar.

I will draw on a biological model of project-selection to illustrate this. Successful projects will attract the developers from non-successful ones and assimilate them to the governance conventions of the project. Since I am modelling competition between governance structures, I will define project success by the mean payoff of the developers constituting the project. I will abstract from heterogenous project sizes, and assume a fixed number of projects of identical size. This can be justified by noting that projects that become fairly large can form subprojects, and develop distinct informal governance norms. However, the heterogeneity and number of projects is not germane to my argument, so I do not explicitly introduce them into the model.

Formally, consider a population of  $K$  projects, each of size  $N$ , thus the state space is now  $X^K$ . Let  $R^K$  be the set of states where every project is in a state in  $R$ , and  $H^K$  to be the state in which every project is at  $n^*$ .

**Proposition 2** *Without project selection,  $H^K$  is the stochastically stable equilibrium.*

Proof: this can be seen by just looking at the long-run distribution of the joint Markov process: the ergodic distribution of the joint process will be just the product of the ergodic distributions of each of the within-project processes. Thus all the states with positive probability in the ergodic distribution will be the product of states with positive probability. So all the recurrent states of the unperturbed process are combinations of  $R$  and  $H$ . We can demonstrate the result by the local resistance test, following (?). It suffices to note that if a state contains an  $R$ -project, the least-resistant exiting edge will have resistance  $R * (R)$ , while the least-resistant entering edge will have resistance  $R * (H)$ . Since there are only two recurrent classes in the within-project game, we know that  $R * (R) = CR * (H) > R * (H)$ , so the least entering edge will have resistance larger than the least exiting edge. Thus, these states cannot be the root of the least-resistant tree, which leaves  $H^K$  as the stochastically stable state.

We will now add project selection. Following Bowles et al. (Bowles and Gintis forthcoming, Bowles, Choi, and Hopfensitz 2003), we randomly choose two projects and convert them both to the state of the project with the highest mean payoff. Let  $D_i(m_1, m_2, \dots, m_K) = |\{m_j | m_j = m_i\}|$ . This means that the between-project dynamic sends the state of the system to  $G(n_1, n_2, n_3, \dots, n_K | m_1, m_2, \dots, m_K) = D_i(m_1, m_2, \dots, m_K) / \binom{K}{2}$  if  $n_i = \max(m_i, m_j)$  and  $n_j = \max(m_i, m_j)$  for some  $(i, j)$  and 0 otherwise.

Now, let us consider the joint process, where the population first experiences within project selection  $A$  times and then between project selection. This is given by the ergodic transition probability matrix  $G^\epsilon(P^\epsilon)^A$ . I claim this has a set of absorbing states  $R^K = R \times R \times \dots \times R$  and  $H^K = H, H, H, \dots, H$ . We can, without loss of generality, assume  $A = 1$ , since applying  $P^\epsilon$  multiple times before applying  $G^\epsilon$  will not change any of the results. This depends crucially on our dynamic being best-response, so one application of the unperturbed transition matrix is sufficient to send the population to a Nash equilibrium. Proof of this follows from the fact  $P$  is idempotent ( $P^2 = P$  or  $P^A = P$ ), and  $\lim_{\epsilon \rightarrow 0} P^\epsilon = P$ . Thus  $\lim_{\epsilon \rightarrow 0} (G^\epsilon)((P^\epsilon)^A) = \lim_{\epsilon \rightarrow 0} (G^\epsilon)(P^\epsilon) = GP$ .

**Proposition 3** *If a stochastic acyclic game has two recurrent classes,  $x_1$  and  $x_2$ , with  $R(x_1) > CR * (x_1)$ , under a best-response dynamic, and project-selection occurs with arbitrary frequency, then if  $K \times CR * (x_1) > R(x_1)$ , project-selection supports the mean-payoff dominant class (assumed to be  $x_2$ ) as the stochastically stable set. The amount of time spent in the mean-payoff dominant class is given by  $O(\epsilon^C R * (x_2))$*

Sketch of Proof: (see Naidu (Naidu 2003) for more details)

Let  $x_1$  be the stochastically stable state in the within-project dynamic. The key is that with exactly two disjoint recurrent classes, it must be that:  $CR * (x_1) = R(x_2)$  and  $CR * (x_2) = R(x_1)$ . That is, the cost of entering one class is the same as the cost of leaving the other. With this in mind, the proof drops out with  $R(x_2^K) = KCR * (x_1)$  since every project must mutate into the basin of  $x_1$  at the same time, otherwise the project-selection will merely return the population to  $x_2^K$  the next round. By assumption  $KCR * (x_1) > R(x_1) = CR * (x_2) = CR * (x_2^K)$ , with the last inequality following because it only requires one project to mutate to  $x_1$  to displace the entire population to  $x_1$  with positive probability.

Invoking this theorem with  $x_1 = R^K$  and  $x_2 = H^K$ , we need  $K > n*$  projects to make the class of states in  $R^K$  stochastically stable.

We now add very mild conformity via state-dependent mutations (Bergin and Lippman 1996, Damme and Weibull 2002). If the population is at a monomorphic state, we then require that mutations occur less frequently, by a power of  $c > 1$ . That is, if an agent would play idiosyncratically with probability  $\epsilon$  at a polymorphic state, they play idiosyncratically with probability  $\epsilon^c < \epsilon$  for small  $\epsilon$ . Conformity only applies at monomorphic states, and  $c$  can be arbitrarily close to 1, we assume it is sufficiently close to one as to not alter the stochastically stable state in the project-selection model. This reflects a common behavioral assumption about humans: it is harder to express innovative behaviors in an environment consisting of a single norm. I am using conformity to select the monomorphic equilibrium,  $A^K$  out of all the possible payoff-equivalent equilibria in  $R^K$ .

**Proposition 4** *With project-selection and an arbitrarily small amount of conformity, the all-Autonomist equilibrium  $A^K$  is stochastically stable.*

Proof: Here, again, I use the method of least-resistance trees, as in Young (Young 1998). Noting that the edge  $R^K$  of the simplex contains the stochastically stable state, we can then note that any tree rooted in the edge will have greater resistance than a tree rooted at the monomorphic equilibrium, since the branch exiting the monomorphic state,  $A^K$ , will have cost  $r_{A^K y} = cr_{xy} > r_{xy}$  for any  $x \neq P$  in  $R^K$  and any  $y \in R^K$ . Note that  $r_{xy}$  is identical across all  $x$  and  $y$  in  $R^K$ . Thus the least rooted tree will be rooted at  $A^K$ . By Young (Young 1998) theorem 3.1, this means that the stochastically stable state is  $A^K$ .

## 5 What About Property Norms?

Why would such governance emerge in software development? Why not in law, economic consulting, or book publishing? Why couldn't projects be managed as a market, with clearly demarcated bounds of ownership preventing conflicts, and agents exchanging money for access to? Why couldn't agents coordinate using some sort of property-rights mechanism to prevent conflict?

This is, in fact, precisely what Raymond argues in "Homesteading the Noosphere". He notes that conflict is resolved via hacker property norms, around project ownership.

"Property is an abstraction of animal territoriality, which evolved as a way of reducing intraspecies violence. By marking his bounds, and respecting the bounds of others, a wolf diminishes his chances of being in a fight that could weaken or kill him and make him less reproductively successful. Similarly, the function of property in human societies is to prevent inter-human conflict by setting bounds that clearly separate peaceful behavior from aggression." (Raymond 1999, pg 139)

People rarely modify or release unauthorized patches or derivative projects without the consent of the project owner, who is generally recognized as the project founder, or legitimate heir thereof. We can capture the effect of well-defined property rights in our model via the bourgeois strategy (Maynard Smith 1982). Bourgeois players use an external signal to coordinate when players should Dominate and when they should just Contribute. Half the time they are the owners in a conflict, and half the time they are not. They defend their turf like Dominators, but are quite happy to contribute unconditionally when they are working on someone else's project. Bourgeois has payoffs given by

<i>strategy</i>	<i>D</i>	<i>C</i>	<i>A</i>	<i>Bourgeois</i>
<i>Bourgeois</i>	$\frac{V-C}{4}, \frac{2V-C}{4}$	$3V/4, V/4$	$\frac{(2-\beta)V-\beta C}{2}, \frac{\beta V+(1-\beta)C+V/2}{2}$	$V/2, V/2$

**Proposition 5** *The Bourgeois strategy is stochastically stable, with or without project selection*

Proof: To show stochastic stability, it suffices to show that Bourgeois is  $1/2$ -dominant. Note also that the payoffs from Autonomists and Bourgeois are equated with 0 contributors and  $V + C/(2V + 3C) < 1/2$  autonomists. Since having any additional Contributors merely improves the payoffs of the Bourgeois (to  $3V/4$ ) while leaving the payoffs of Autonomists unchanged, this implies that Bourgeois is a best-response to any strategy that at least  $1/2$  probability on Bourgeois. Thus the Bourgeois strategy is  $1/2$ -dominant. Thus, within group, the Bourgeois equilibrium is stochastically stable. We ignore the Dominator strategy because at any absorbing state where Dominator can do well, Bourgeois can do better. However, since the mean group payoff is the same under the Bourgeois or Punisher equilibria, project selection will not alter the stochastically stable state. Also, both are monomorphic equilibria, so conformism will not alter the relative transition costs.

Property rights can be used to effectively mediate some types of conflict, and the model predicts that we should see extensive use of informal property rights as an instrument of governance. The next section, however, should illuminate why most conflicts cannot be resolved via property rights.

## 5.1 Limited Raymondism and Contested Bourgeois

While it is true that there are informal property rights in OS projects, they are responsible for an coordinating an incredibly small portion of the myriad interactions that characterize successful Open-Source projects. It is quite rare to have someone unilaterally assert control over major decisions over a project on the grounds that it is "theirs". While some conflicts are clearly resolved by deference to ownership, the overwhelming number of debates and arguments that go on within projects are documentation for the importance of collective governance. Evidence against the Raymond perspective on Lockean property rights is that people *talk* about production. OSS projects look little like independent producers exchanging nuggets of code in exchange for kudos in a C++ comment or README file. Instead, the process is actively social with people discussing what to do, what's needed, who is unpopular, etc. The prosocial nature of the interactions are a hint that agents are not merely mediating their interactions through an established set of property norms, but rather forming a community of peers.

Why aren't all these issues resolved via ownership, with decisions made by the de jure project leader? A reason is that the conflicts are often over technical issues, so it is important to air multiple viewpoints and expertise from people who may not be the project owner. If the task being debated is quite technical, and knowledge is sufficiently distributed, project ownership would be a poor arbiter of disagreements. It may very well be that a project non-owner has the best information regarding how to solve a particular problem, and therefore their input and decision-making ability is to be valued. Most innovation in OSS projects is collective invention, in the sense of Allen (Allen 1983); agents share their innovations, allow additional improvements, and do not rely on patenting or copyright to benefit from their inventions. Not making technical decisions collectively could be highly detrimental to the project's success.

If property rights were really the basis for conflict resolution, no conflicts would ever manifest, as agents would unconditionally contribute whenever working on someone else's project. But rarely are technical discussions settled by fiat; the Compiere project has had over 1400 posts to a technical discussion about database independence. Similarly, when projects like Debian decided to incorporate as non-profits, the decision was not unilaterally undertaken by the project owner (OMahony 2003), but instead was talked about within the project community for some time. Thus, Open-Source projects resemble Wright and Fung's (Wright and Fung 2003) deliberative communities, where agents discuss the best course of action collectively, without appeal to their private interest. Rarely are arguments made that begin with "it would be best for my personal interest if...".

Thus, if property rights were the basis for decision making, there would be a fair number of conflicts due to the lack of a close correlation between project ownership and expertise. To incorporate this into the model, we introduce Contested Bourgeois, who is only able to regulate a fraction  $1 - \mu$  of potential conflicts via

property rights. Thus the Contested Bourgeois winds up mistakenly flaming/forking a substantial fraction of the time. The payoffs to Contested Bourgeois are given by:

<i>strategy</i>	<i>D</i>	<i>C</i>
<i>Bourgeois</i>	$(1 - \mu)V/4 + (\mu)(v - c/4), (v - \mu C)/2 + (1 - \mu)(v - c)/4$	$V/2 + 1/2((1 - \mu)V/2) + \mu V), (1 + \mu)V/4$ (1 -

**Proposition 6** *There is a  $\mu^* = \frac{V}{2C-V}$  such that if  $1 > \mu > \mu^*$  then the population spends no time at the Contested Bourgeois equilibrium, and the Rousseauian equilibrium is stochastically stable.*

Proof: Note that Contributors dominate Bourgeois if  $(1 - \mu)V/4 > V - \mu C/2$ , which is true for  $\mu$  sufficiently high. Contested Bourgeois will be invaded by Contributors, and so will no longer be an absorbing state of the unperturbed dynamic.

## 6 Forking: Some Anecdotal Evidence

In the early 1990s, Bill and Lynne Jolitz, veterans of the Berkeley Unix effort, finished a port of the Unix-based operating system to the Intel 386 architecture. He then "simply put it up on for anonymous FTP and let anyone who wanted it download it for free. Within weeks he had a huge following." (Debona, Ockman, and Stone 1999). Jolitz, maintaining a full-time job, was unable to keep up with the patches and suggestions that were pouring in from the user base. Jolitz's insistence on personally reviewing every piece of code, and not releasing before he had prevented a large number of contributors from agglomerating.

It is widely acknowledged that 386BSD preceded Linux by a non-trivial margin, and even Linus Torvalds has noted "If 386BSD had been available when I started on Linux, Linux would probably never have happened." (Salon.com, may 17, 2000). However, it did not have the rapid re-release schedule that Linux did, and failed to elicit a large amount of community support, partly, I would argue, as a result of an unwillingness to share responsibility and rights with the community. Eric Raymond notes that the primary difference was in the social organization of the project. While Torvalds "released early and often", the Jolitzes micro-managed the evolution of 386BSD. A large cluster of potential contributors, frustrated with the tardy incorporation of their patches and suggestions, created NetBSD, possibly the first Open-Source operating system fork ever. NetBSD consisted of the Jolitz's 386BSD operating system along with a bundle of patches and additions designed to improve the system.

The moral of this story is that governance matters for the success of Open-Source projects. Torvald's rapid responses to patch and design contributions stand in stark contrast to the "Cathedral" style of the Jolitz's. The informal egalitarianism of Linux triumphed over the autocratic style of 386BSD, despite the formal similarities in their organizational structure. There are numerous other instances of forks occurring over disagreements in style, conflict with management, and other conflicts, for example the Harbour project, which officially split into the xHarbour and Harbour projects in August 2001. As the xHarbour website puts it: "xHarbour was established to serve as a more aggressive alternative to [Harbour's] conservative style of development"<sup>1</sup>.

Another interesting fork occurred in the Samba project. Consider the following news story from ZDnet: "The company made the decision to fork its code base in order to appease two warring camps. One project, those behind the newly minted Samba TNG (for "the next generation"), wanted to provide support for new printer-and-networking functionality built into Windows 2000. The other was less interested in supporting some of the newer, less standards-compliant Microsoft( msft) add-ons, such as Microsoft's customized version of Kerberos. "We need this like we need a hole in the head. Yet another camp of developers who can't/haven't learned to play well with others. 'I'm taking my marbles and going home, bwaaahhh,' " said poster arothstein. "Open source seems to be a great forum for encouraging childish behavior."<sup>2</sup>.

<sup>1</sup> [www.xHarbour-project.org](http://www.xHarbour-project.org)

<sup>2</sup> <http://zdnet.com.com/2102-11/2-524722.html>, accessed feb 17th, 2004

When the Israeli government was considering adopting Linux, economist Robert Sauer wrote an article in the Israeli business paper *Globes* noting that "The forking of open-source projects occurs when passionate disputes between open-source software developers over product design lead to the splintering of projects into a multitude of varieties. With proprietary software, forking generally does not take place since development is centralized within a firm and disciplined by market forces."<sup>3</sup> Sauer is right about the fact that forking cannot happen in a firm, but incorrect about the frequency of forks. They are relatively uncommon, which does not diminish their importance as an instrument for collective enforcement of norms.

one poster on Slashdot noted in response:

"Sometimes forking can hurt a project, but often times it encourages innovative work in a different direction. Usually, however, it signifies a problem in the management of the project; if a developer is frustrated by the project leadership, they might fork the project rather than struggle to get their viewpoint heard on the main project. One of the testaments to the managerial skills of Linus Torvalds and his lieutenants is the fact that the Linux kernel has not undergone major forking. Kernel developers in general feel that they can get their work done adequately on the main Linux branch."<sup>4</sup>

These illustrations are to show that forking is a real possibility, occurs in response to conflicting opinions, and is often exercised collectively. It is a collective sanction, along with flaming/insulting, so essential to Boehm's account of early human egalitarianism, and the persistent threat of it restrains would-be despotic developers from acquiring any real authority or violating project norms. A reader might claim that, in fact, flaming and shunning constitute the primary mechanism of norm enforcement. Extreme shunning, however, is functionally identical to a fork by the overwhelming majority of the project community. Another point is that flaming only matters within an established community, since insults only matter if they are delivered by someone who is recognized by the insulted. Otherwise, leaders would not be overly bothered by the jeers from strangers that accompanied selfish decisions.

A key mechanism for generating the credible threat of forking, however, lies in the conjunction of the GPL and the non-rival nature of software. By forcing developers to contribute the modified code back to the community, the GPL guarantees the ability of any developer to attempt a fork. Like the spear in foraging societies, the ability to fork, especially when wielded collectively, is decisive for promoting egalitarian governance and sharing of knowledge and skills. Those that refuse to share find themselves alone in the sandbox. The GPL, by legally enshrining the right to fork code, forces projects to develop prosocial norms of governance, in order to economize on conflict. This is, in some sense, deliberating using a type of property rights to "crowd in" egalitarian governance. But note that the non-rivalness of the code is essential to this feature; the ability to fork is only credible because there is freely available code to copy and start a new project. In some sense, the right to theft is enforced by law.

Another factor encouraging this type of governance is the environment in which the cultural roots of OSS sprouted. In the mainframe-terminal computer environment of the 1960s -early 1980s, the problems faced were not solvable by market/command means. These problems included the allocation of time-slices on the mainframe, sharing of applications and code, as well as other public-good and common pool resource allocation problems. In the absence of strong property rights and market signals, as well as a distinct lack of central authority, it is well documented (Ostrom 1990) that prosocial norms are likely to develop.

The forking threat imposed by the volunteer labor market and project competition for developers is the underlying political apparatus for the rank-and-file developers to secure respect and submission from the leaders. Lerner and Tirole identify a coordinating role for leaders such as Linus, but fail to recognize the direction of authority. McGowan's proposed puzzle is a non-issue; there is no political authority worth speaking of, so explaining why people voluntarily submit to it is not a problem.

---

<sup>3</sup><http://www.globes.co.il/serveen/globes/DocView.asp?did=747399&fid=980>, accessed feb 17, 2004

<sup>4</sup><http://slashdot.org/article.pl?sid=03/12/09/149224>, accessed feb 17, 2004

## 7 Conclusion and Future Work

While formal econometric testing of the hypothesis put forward here is difficult, by nature of the questions asked, it needn't be impossible. Future work will attempt to use statistical tools from anthropology and sociology, especially network analysis, to determine whether in fact OSS projects are relatively egalitarian in the distribution of social dominance. However, I should note that the infrequency of forking is not evidence against the model put forward here, because in equilibrium forks almost never occur.

Besides explicitly modelling multi-level selection of conventions, this paper makes two new observations. Firstly, there is a substantial amount of collective deliberation within OSS projects, and project owners/leaders/managers are held accountable to project norms, via the threat of forking and the pressure of competing for volunteer developers. Project competition for developers forces them to evolve governance mechanisms that minimize potentially costly forks. Secondly, Eric Raymond's claims about informal property rights as a mode of conflict resolution are just a small part of the governance apparatus existing within OSS projects, since the dispersion of expertise makes project owners suboptimal autocrats even in their own projects.

This paper should be seen an initial step towards a larger project to deploy evolutionary institutional economics to understand the complex matrix of interactions that the Open-Source phenomena has thrown onto the social science table. The conventions of governance seem like a natural place to wed the tools of evolutionary game theory with the substantive claims about egalitarianism often debated in Open-Source circles. The appearance of these conventions should be seen as an emergent property of the institutions and behaviors of the developer community; competition for volunteer developers, developer preference for egalitarianism and autonomy, the non-rival nature of the assets and the legal architecture provided by the GPL come together to generate these unique governance conventions. The robustness of these norms, particularly whether or not they will weather the increasing involvement of businesses and government, remains to be seen.

## 8 Appendix

### 8.1 Heterogeneous Agents

The high inequality of contributions, for example, are manifest in both big-game hunting as well as the provision of elegant, functional code, as the below graphs illustrate from the FLOSS study shows. I hope to argue that the famous political egalitarianism, properly understood, that exists in foraging communities, also applies to Open-Source projects. Hence the subtitle of the section.

Consider the same model, except that we add a small fraction, say  $\delta$  of highly productive agents. The prize in games against these agents is  $V_1 > V$ . This implies that there is a lower threshold of Contributors necessary to induce these agents to switch to Dominator, call this  $m_1^* < m^*$ .

Formally, let the state space of the Markov process now be  $X(p, q) \times P, G, S$  for simplicity we only allow pure strategies in the idiosyncratic population. Thus the unperturbed dynamic takes  $p, q, \sigma$  to  $BR_1(p, q, \sigma), BR_2(p, q, \sigma)$

if  $1 - p = q < m_1^*$  then  $BR_1(p, q, P/S) = p, q, BR_2(p, q) = P/S, BR_1(p, q, G) = (1, 0)$

if  $1 - p = q \in (m_1^*, m^*)$ , then  $BR_2(p, q, P) = BR_2(p, q, S) = G$

if  $1 - p = q \in (m_1^*, m^*)$  and  $\sigma = G$ , then  $BR_1(p, q, G) = (1, 0)$

if  $1 - p = q > m^*$  then  $(BR_1(p, q, *), BR_2(p, q, *)) = (0, 1 - V/C, G)$

Intuitively, what happens when we introduce these agents is that they become Dominators with a lower threshold of sharers. This makes the other sharers instantly realize they need to return to being an autonomist, and thus mitigates drift away from the Rousseauian equilibrium. The interesting thing about this model is that the inequality in productivity is the element that stabilizes egalitarian governance. This is a

feature of both foraging cultures and open-source communities, and, I would argue, many other governance structures.

Proposition: Sufficient idiosyncratic inequality (high  $V_1$ ) can stabilize an autonomist Nash equilibrium.

Proof: Break this up into 3 parts: Claim 1) there exists a  $a^* \leq \bar{a}$  that is the threshold level of altruists for high productivity agents to switch to Dominators. Proof, obvious.

Claim 2) The unperturbed dynamic takes this state back to the all-A state costlessly, for every state between  $a^*$  and  $\bar{a}$  Proof, follows from above Markov process, albeit with some work.

Claim 3) this makes  $R(0, a^*) > CR^*(0, a^*)$ , meaning that  $(0, a^*)$  is stochastically stable. follows since  $a^* - \bar{a} > n^*$

where  $n^*$  solves  $\pi^P(n^*/N, (1 - n^*/N)(1 - V/C)) = \pi^G(n^*/N, (1 - n^*/N)(1 - V/C))$

An interesting consequence of this is that competition, heterogeneity (see Appendix), and self-interest conspire and interact together to generate egalitarian institutions. Inequality of talent, an optimizing dynamic, and competition for volunteers acts to stabilize the all-Punisher equilibrium, where agents who try to dominate the others are socially sanctioned. This is contradictory to the relationship between endowments and authority that occur in markets. Easwaran and Kotwal (1984), Bowles (2004), Ragan and Zingales (2002), all note that ownership of productive assets translates into power or residual claimancy within the firm. OSS projects are fairly different; having a productive asset/skill in a competitive environment does not translate into power or residual control.

()

## References

- ALCHIAN, A., AND H. DEMSETZ (1972): "Production, Information Costs, and Economic Organization," *American Economic Review*, 62, 777–795.
- ALLEN, R. (1983): "Collective Invention," *Journal of Economic Behavior and Organization*, 4(1), 1–24.
- BENKLER, Y. (2003): "Coase's Penguin, Linux and the Nature of the Firm," *Yale Law Journal*, 112.
- BERGIN, J., AND B. LIPPMAN (1996): "Evolution with State Dependent Mutations," *Econometrica*, 70, 281–297.
- BOEHM, C. (1999): "The Natural Selection of Altruistic Traits," *Human Nature*, 10(3), 205–252.
- BOWLES, S. (2004): "Economic Behavior and Institutions: An Evolutionary Approach to Microeconomics," Princeton University Press.
- BOWLES, S., J. CHOI, AND A. HOPFENSITZ (2003): "The Co-evolution of Individual Behaviors and Social Institutions," *Journal of Theoretical Biology*, 223, 135–147.
- BOWLES, S., AND H. GINTIS (1988): "Contested Exchange: Political Economy and Modern Economic Theory," *American Economic Review*, 78(2), 145–150.
- (forthcoming): "The Evolution of Strong Reciprocity: Cooperation in Heterogeneous Populations," *Theoretical Population Biology*.
- DAHL, R. A. (1957): "The Concept of Power," *Behavioral Science*, 2, 201–215.
- DAMME, E. V., AND J. WEIBULL (2002): "Evolution in Games with Endogenous Mistake Probabilities," *Journal of Economic Theory*, 106, 296–315.
- DEBONA, C., S. OCKMAN, AND M. STONE (1999): *Open Sources*. O'Reilly, Sebastol, CA.
- ELLISON, G. (2000): "Basins of Attraction, Long-Run Stochastic Stability, and the Speed of Step-by-Step Evolution," *Review of Economic Studies*, 61, 567–588.
- HEALY, K., AND A. SCHUSSMAN (2004): "The Ecology of Open-Source Software," *University of Arizona Working Paper*.
- KANDORI, M. G., G. MAILATH, AND R. ROB (1993): "Learning, Mutation, and Long Run Equilibria in Games," *Econometrica*, 61, 29–56.
- KAPLAN, H., AND J. LANCASTER (1984): "The Evolution of Reproductive norms," *Current Anthropology*, 25(1), 113–115.
- LERNER, J., AND J. TIROLE (2000): "Some Simple Economics of Open Source," *Journal of Industrial Economics*, 52, 197234.
- MAYNARD SMITH, J. (1982): *Evolution and the Theory of Games*. Cambridge University Press, Cambridge, UK.
- MCGOWAN, D. (2001): "Legal Implications of Open-Source Software," *University of Illinois Law Review*, 1, 241304.
- MOODY, G. (1991): *Rebel Code*. O'Reilly, Sebastol, CA.
- NAIDU, S. (2003): "Altruism is a Stochastically Stable State," University of Massachusetts-Amherst.
- OMAHONY, S. (2003): "Guarding the commons: how do community managed software projects protect their work?," *Research Policy*, 32(7), 1179–1198.

- OSTROM, E. (1990): *Governing the Commons: The Evolution of Institutions for Collective Action*. Cambridge University Press, Cambridge, UK.
- RAYMOND, E. (1999): *The Cathedral and the Bazaar*. O'Reilly, Sebastol, CA.
- ROSSI, A. (2004): "Decoding the Open-Source Puzzle," University of Siena Working Paper.
- WRIGHT, E., AND A. FUNG (2003): *Deepening Democracy*. Verso, New York.
- YOUNG, H. P. (1998): *Individual Strategy and Social Structure: An Evolutionary Theory of Institutions*. Princeton University Press, Princeton, NJ.