Open Source Software Production, Spontaneous Input, and Organizational Learning

Garzarelli, Giampaolo and Fontanella, Riccardo

Institutions and Political Economy Group (IPEG), School of Economic and Business Sciences, University of the Witwatersrand

2010

Open Source Software Production, Spontaneous Input, and Organizational Learning

Giampaolo Garzarelli (Corresponding author)
Institutions and Political Economy Group (IPEG)
School of Economic and Business Sciences
University of the Witwatersrand
Private Bag X3, WITS 2050
Johannesburg, Republic of South Africa
+27.11.717.8128 (Tel.)
+27.11.717.8081 (Fax)
Giampaolo.Garzarelli@Wits.ac.za

Riccardo Fontanella
Institutions and Political Economy Group (IPEG)
School of Economic and Business Sciences
University of the Witwatersrand
Private Bag X3, WITS 2050
Johannesburg, Republic of South Africa
fontiric@gmail.com

Forthcoming *American Journal of Economics and Sociology*.

Acknowledgement: We are indebted to Lyndal Keeton and two anonymous referees for their very valuable feedback.

Open Source Software Production, Spontaneous Input, and Organizational Learning

Abstract

This work shows that the modular organization of voluntary Open Source Software (OSS) production, whereby programmers supply effort of their accord, capitalizes more on division than on specialization of labor. This is so because voluntary OSS production is characterized by an organizational learning process that dominates the individual one. Organizational learning reveals production choices that would otherwise remain unknown, thereby increasing productivity and indirectly reinforcing incentives to undertake collective problem solving. (71 words.)

Key Words

Division of Labor, Mistake-ridden Learning, Modularity, Open Source Software, Self-selection, Voluntary Production

JEL Codes

D20, L17, L23

1. Introduction

With success stories such as Apache, Linux and Mozilla, Open Source Software (OSS) has entered into the vocabularies of millions of individuals worldwide. The increasing number of OSS users[1] has stimulated a vast and growing interest on the economics, and more generally social science, research front.[2] What seems to puzzle most of this research is that, contrary to more familiar economic theory, the development method of OSS should not have accounted for its success. How can a number of individuals dispersed around the world who mostly rely on open standards and an ethos of code sharing lead to a stable production process? Thus OSS production is perceived to be at odds with many traditional production paradigms (Feller and Fitzgerald 2002). This is especially the case for OSS development that is *voluntary* in nature.

Voluntary OSS development – the principal interest of this paper – is in fact a production process whereby individuals supply their input of their own accord. That is to say that it is a process where there is mostly self-selection in – rather than direction of – task performed (Langlois and Garzarelli 2008). By letting individuals contribute effort according to their own volition, voluntary OSS development tries to profit from the "distributed intelligence" of members of virtual communities (Kogut and Metiu 2001). That is, to maximize the gains from the creation, reuse, and trade of one factor: knowledge (Garzarelli *et al*. 2008). This "mindshare" approach has been referred to also as "collective invention" (Osterloh and Rota 2004).

In more ways than one, then, voluntary OSS production can be conceived of as another tangible illustration of the advantages that can emerge from the spontaneous interaction of many individuals each possessing limited knowledge and

---

1 See for example the regularly-updated statistics freely available on Netcraft: http://news.netcraft.com/.

2 One reference for all: the MIT website that collects OSS papers, http://opensource.mit.edu/.

pursuing their own interests (Hayek 1937, 1945; Jensen and Meckling 1992; Raymond 2001). In other words, voluntary OSS production is a contemporary illustration that shows how "the productivity of social cooperation surpasses in every respect the sum total of the production of isolated individuals" (Mises 1960, p. 43).

But this poses the question of the origin of the emergent benefits of voluntary OSS development. By considering production as a set of interrelated tasks rather than as a mere technological relationship[3], we offer a counterintuitive answer to this question, namely, that in the main the emergent benefits of voluntary OSS production do not derive from specialization but from division of labor. The division of labor of voluntary OSS production is not ordered sequentially as specialization would require, but rather in a parallel and overlapping form. And the benefits from this parallel and overlapping division of labor, it is suggested, trump those of specialization. This is so because in voluntary OSS production the learning by doing is primarily organizational rather than individual.

To express our claim in other terms, voluntary OSS production is a social learning process that gives off signals that reveal problems regarding production relationships and their inputs and outputs. The need to correct problems acts as a mechanism that stimulates internal urges, compelling individuals to seek out new problem-solving techniques. These incentives that are awakened by problems ultimately also increase productivity, because they reveal production choices that would otherwise remain unknown.[4]

---

[3] On which see, for example, Georgescu-Roegen (1970), Winter (2005), and Dosi and Grazzi (2006).

[4] Cf. Rosenberg (1969).

2. Modularity and two divisions of labor

Before exploring the rudimentary building blocks of the division of labor of voluntary OSS production, it is useful to quickly do two things. The first is to introduce the rules governing voluntary OSS division of labor. Without at least a basic awareness of these rules it is difficult to convey a sense of how a parallel and overlapping process of production can work. The second is to shed some light on the specific characteristics that define vertical and horizontal divisions of labor, i.e., the two divisions of labor that Smith (1981[1776]) obliquely alludes to (Leijonhufvud 1986).[5] As will be clear before long, in fact, vertical and horizontal divisions of labor are two useful heuristic expedients that will help us to more easily understand how voluntary OSS production relates to more familiar structures of production.

2.1. Modularity rules

Software production is an activity that is very knowledge intensive: it embodies the knowledge of many programmers, each of whom only knows a portion of what others know. As such, it manifests useful give-and-take: software programming is a *social* learning process (Baetjer 1998). But in order for the learning to be sufficiently coherent with the overall aims of a software project, there must be some basic rules in place that channel it in the right direction. This is especially so in cases of voluntary OSS production where input is spontaneous.

Most voluntary OSS projects rely on the rules of modularity (Simon 1998[1962]). As the name implies, modularity is about breaking up a system into parts (modules) in the attempt to make it more manageable. By making a project more manageable through decomposition, modularity assists the division of labor (e.g.,

[5] In a recent contribution, Leijonhufvud (2007) returns to these themes, but for some reason inverts his original classification, calling the horizontal division of labor vertical and, by difference, implying that the vertical division of labor is the horizontal. Here, we will stick to the original 1986 definitions.

Baldwin 2008). And to leverage from divided labor, modular decomposition minimizes modular interdependencies by hiding information. Instead of all knowledge being communicated across modules, information hiding lets modules keep some of their knowledge 'secret' from the rest of the project. In this fashion, no module can interfere with the data and functions of other modules. Knowledge is thus *encapsulated* within modules, and a programmer need not necessarily hold any information about the other modules of the project with which his module interacts (Parnas 1972). More generally, information hiding is important because it "allows developers to understand the system better by viewing it at a high level of abstraction" (Baetjer 1998, p. 107).

However, given that separate modules form part of the same system, communication among modules cannot be entirely blocked. Or, more precisely, as long as we acknowledge that the system is directed towards a goal, as is for instance the case of a software project, the system cannot be completely decomposable, but only in part. Modules need to know what their functions are as well as the functions of *complementary* modules (the *architecture* of the system), the nature of their relationships with their complementary modules (the *interface* of the system), and their performance relative to other modules (the *standards* of the system). Baldwin and Clark (2000) refer to these modular properties as the *visible design rules* of a modular system. Therefore, a modular system directed towards an end is one that should be *nearly* decomposable, preserving the possibility of cooperation among complementary modules by sharing and communicating visible design rules, while at the same time preserving the *hidden* design parameters specific to each module (e.g., Langlois and Garzarelli 2008).

The distinguishing mark of modularity is then the forcing, as it were, of the use of specialization by relying on rules that blind irrelevant information. In the case of voluntary OSS production this is also the case. However, voluntary OSS entails that, in addition to being able to spontaneously contribute to tasks that reflect one's specialization, a programmer is able to spontaneously contribute to tasks for which he or she is non-specialized. Consider the following system-versus-individual metaphor to more finely hone this observation. At the system level, individuals are specialized in the sense of being programmers; but at the individual level, each programmer may not always contribute according to his primary specialty.[6] And it is because of the existence of this imperfect matching that specialized effects turn out to be secondary vis-à-vis divided ones voluntary OSS production. In many ways, the rest of this work can be interpreted as an attempt to elaborate this point, that is, to begin to direct attention to the division of labor properties of one type of modular organization.

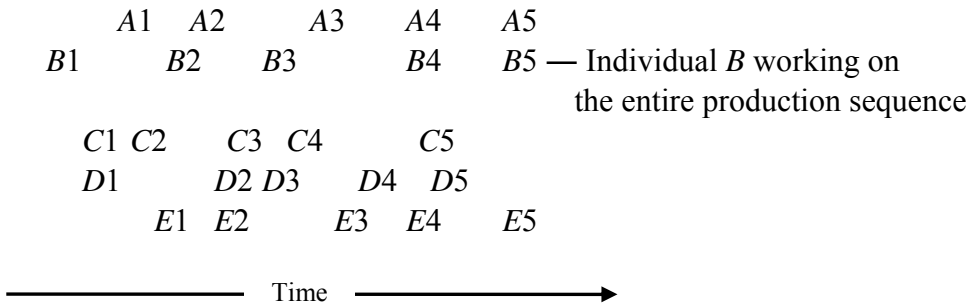
2.2. Vertical and horizontal divisions of labor

A vertical division of labor takes place when an individual performs each of the tasks of a process of production of a particular good. Consider, to illustrate, a carpenter who needs to produce a piece of furniture. The carpenter's tasks, to name a few, would include selecting the appropriate wood, cutting and gluing the wood to design, treating the wood and often even selling the furniture once completed. Figure 1 illustrates this vertical division of labor where different individuals (*A*, *B*, *C*, *D*, *E*) perform every task (1, 2, 3, 4, 5) independently. Thus, under vertical division of labor

[6] We may even make the metaphor slightly more precise. If we define, following Ames and Rosenberg (1965), specialization as the reciprocal of skill, we have the ratio of the number of doers (individuals) per activity (task). We can accordingly define a specialization index that lies between 0 (complete non-specialization) and 1 (complete specialization). So, in terms of this index, in voluntary OSS production it is not uncommon in any point in time to have several individuals whose specialization index is significantly below 1 in at least one task performed while having an overall index of 1 in the primary activity, namely, programming.

an individual boasts a wide repertoire of skills, implying a lack of full specialization. That is to say that vertically divided individuals are not committed to performing solely one task according to an opportunity cost-minimizing criterion, but rather perform a multiplicity of related tasks according to both absolute and comparative advantages.

Figure 1: Vertical division of labor



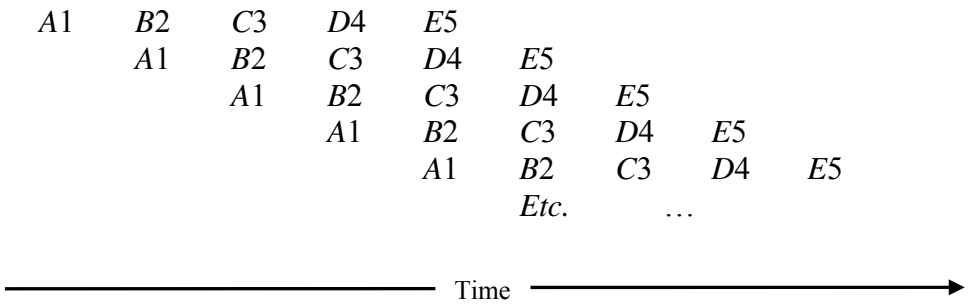
Source: Leijonhufvud (1986, p. 208).

But from Smith (1981[1776]) we also know that the division of labor is constrained by the extent of the market. As the market for a particular product expands, the tasks involved in its production process can develop into specialized ones. When such a demand is reached, an individual is able to dedicate himself solely to one task. The result: horizontal division of labor, a situation typical of the factory system, and later at the root of the modern corporation (Leijonhufvud 1986). As Georgescu-Roegen (1970, p. 8) phrases it, "the increased specialization of labor could not have come about unless an increased demand had already induced most craft shops to introduce the system line. There can be little doubt about it: the factory system was born in an artisan's workshop, not in a factory."

Under a horizontal division of labor, market demand is sufficient to support an individual's commitment to merely one task of a production process. By implication, such an individual is able to become more refined at that single task. But since there is

no free lunch, such specialization comes at the expense of a contracting repertoire of skills. As Knight (1967, p. 21) vividly describes it, "it is especially significant that the most important source of gain" – specialization – "also involves the most important human cost," viz., the narrowing of one's "personality." Contrary to the vertical, then, the horizontal division of labor implies that individuals are specialized. Figure 2 illustrates a horizontal division of labor in terms of our previous notation. We readily see how each individual performs only one task of the sequence (*A* performs just 1, *B* performs just 2, etc.).[7]

Figure 2: Horizontal division of labor



Source: Leijonhufvud (1986, p. 209).

3. Voluntary OSS division of labor: A stylized model

Open source guru Raymond (2001) reminds us that in voluntary OSS production, as elsewhere, innovation and discovery often are a direct, if unintentional, product of the satisfaction of desires. At times, there may be a specific need that an individual wishes to satisfy (e.g., adapting a software package to a new printer). In these cases, the individual is "intrinsically motivated," that is, he freely sorts himself into some task that he desires to perform. Intrinsic motivation usually derives from work that is considered interesting, and can be "crowded out" if an individual senses, for example,

[7] Compare Houthakker (1956) for a more complete cost-benefit analysis of specialization.

that he is being monitored or supervised (Frey 1997). As Deci (1971, p. 105) originally put it, if "external rewards are given for an intrinsically motivated activity, the person perceives that the locus of control or the knowledge or feeling of personal causation shifts to an external source, leading him to become 'a pawn' to the source of external rewards. Similarly, … external rewards affect the person's concept of why he is working and his attitude toward the work."

And yet, not all outside actions crowd out intrinsic motivation. When an outside action is perceived to be a controlling one, such as in the case of a principal-agent relationship where the principal takes over some of the agent's actions, we are most likely to see a drop in intrinsic motivation. When an outside action instead is perceived as informative, such as in the case of positive reinforcement to the agent from the principal, we are most likely to see no drop in intrinsic motivation or perhaps even a rise in it (Frey 1997, p. 432). As illustrated below, in the case of voluntary OSS production, where, with few exceptions, we may liken everyone to his own principal, an important source of an outside action that is informative is learning from the errors and contributions of others.

At the same time, however, the capacity to perform a self-selected task to satisfy a specific need is also a function of the number of task(s) accessible to the individual. When the tasks at an individual's disposal are limited, the ability to satisfy a need is hindered. Think about the quintessential firm where usually not everyone is working on his or her preferred task in every point in time. In such types of organization, the incentive to pursue the satisfaction of a particular need is often abandoned. But when tasks are not limited the means of reaching satisfaction are extended. When this is the case, the incentive to pursue the satisfaction of a particular need increases. "The more extensive the agents' participation possibilities are, the

higher is the work morale" (Frey 1997, p. 431). Voluntary OSS organization presents a potentially unlimited set of production tasks that one can freely align to.

In voluntary OSS production individuals therefore have a high intrinsic motivation because the tasks instrumental in satisfying their specific needs, such as customizing a particular software program, are open to them. As a matter of fact, in many cases they themselves create the task in order to solve a particular puzzle that is bugging them. Indeed, this is Raymond's (2001, p. 23, emphasis removed) first important lesson about voluntary OSS programming, namely, that "[e]very good work of software starts by scratching a developer's personal itch." As a result, there is a strong incentive behind these voluntary contributions; and a strong incentive behind a contribution means that there is greater scope for productive behavior. It is exactly this pool of self-motivation, this congeries of "spontaneous incentives" emerging from individual needs and desires, that voluntary OSS production attempts to seize. Raymond (2001) actually expresses surprise after discovering that there are tremendous productivity gains accruing through such software "development style" and employs two useful ideal types to analyze it: the cathedral (centralized) and the bazaar (decentralized).

Just like the employees in the quintessential firm, the cathedral mode of production proposes that individuals be assigned to tasks on the basis of their competence. In this way, it is suggested, knowledge can be directed to its most productive use. Consequently, a cathedral approach gives rise to a centralized development structure reminiscent of a horizontal division of labor. Proprietary software echoes this form of economic organization: the development process is left to a limited group of highly specialized programmers. However, Raymond (2001, p. 8) points to a significant shortcoming of this: in "a cathedral-builder view of

programming, bugs and development problems are tricky, insidious, deep phenomena. It takes months of scrutiny by a dictated few to develop confidence that you've winkled them all out." This is why, to this day, proprietary software takes quite some time to release.

Conversely, the bazaar assumes that a limited set of consciously organized individuals will never completely possess all the necessary knowledge to always solve software production problems. As Hayek (1945, p. 519) famously wrote, knowledge "of the circumstances of which we must make use never exists in concentrated or integrated form, but solely as dispersed bits of incomplete and frequently contradictory knowledge which all separate individuals posses." With each individual only possessing limited and different knowledge, it would seem impossible to assign tasks to a set of individuals in a manner that knowledge always would be put to its most valued use. Besides knowledge being dispersed and idiosyncratic, there is another reason as to why this is so: the growth of knowledge. Knowledge is about dynamic organization, that is, about changing interrelations among qualitative patterns of stimuli that belong to, and can change, a system. It is not about passive quantitative patterns of stimuli that just serve a system without affecting it (e.g., Fransman 1994; Langlois and Garrouste 1997): "that which we call knowledge is primarily a system of rules of action assisted and modified by rules indicating equivalences or differences of various combinations of stimuli" (Hayek 1978, p. 41). So how does one try to improve the allocation and use of knowledge at every point in time when such knowledge is not identical, not fully endowed to everyone and not constant? Advocates of voluntary OSS place their chips on decentralization. And evidence shows this bet to pay off (Giuri *et al.* 2010).

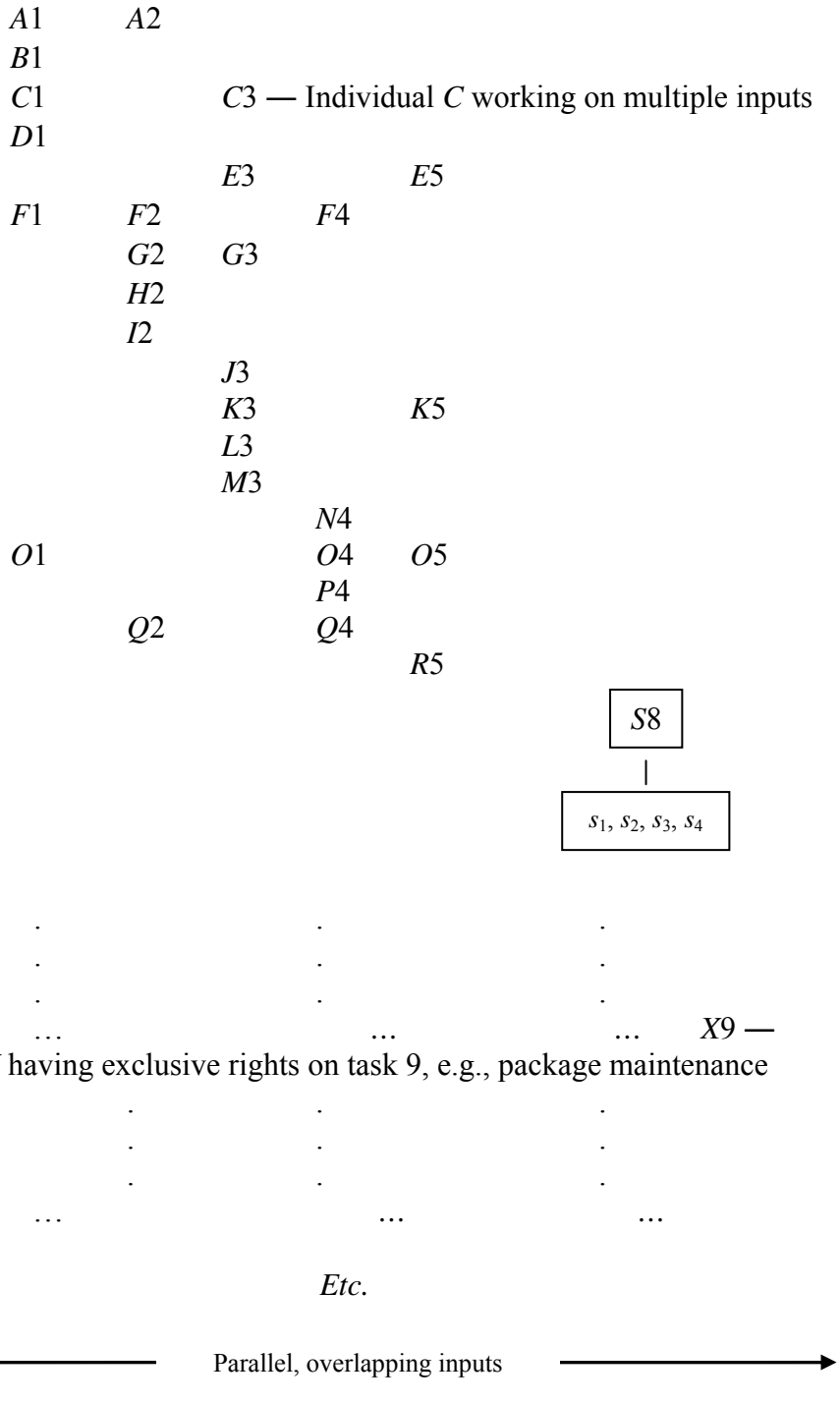
A bazaar organizational form attempts to leave all input options open by leaving the production process open. The production process is open in the sense that individuals are not necessarily assigned to, and hence not necessarily restricted to, tasks based on their implied specialization. Rather, the bazaar favors voluntary collaboration, i.e., it promotes the spontaneous convergence of distributed knowledge. Given a very large set of decision makers collaborating through the exchange of knowledge, it is possible to identify and formulate a wider array of problems and to find a larger set of alternative solutions. This is what Raymond (2001, p. 8) refers to as "Linus's Law": "given enough eyeballs, all bugs are shallow," or, simply put, what one pair of eyes might miss, there are countless other pairs that can make up for it. In sum, the freedom to access, copy and modify source code is a way for voluntary OSS production to tap into a larger collective intelligence that enables to benefit from a broader problem-solving ability.

But this begs the question of what kind of division of labor voluntary OSS production engenders, namely, what are the division of labor dynamics typical of the bazaar? As hinted, voluntary OSS development leaves all input options (i.e., production tasks) open, which in turn implies that programmers have the liberty to self-select into any production task(s) they desire. In actual fact, this means that in such a production setting individuals are not bound to tasks that correspond to their primary specialization. The upshot is that in any point in time it is possible for any number of individuals to be working on single or multiple tasks *irrespective* of specialization.

Figure 3 is a stylized model that tries to capture the essence of such voluntary OSS division of labor. It shows how task 1 is undertaken by a number of individuals, for example individuals *A*, *F* and *O*, whom also can be seen contributing towards

production tasks 4 and 5. Similarly, task 3 is also undertaken by numerous individuals of whom some (*C*, *E* and *G*) are contributing towards multiple tasks.

Figure 3: Voluntary OSS production



At the same time, within a bazaar it is possible that certain production tasks be exclusively reserved for and performed by selected individuals. This is especially

evident in the early stages of an open source project when the process of production is not yet fully defined (Murdock 2003; Langlois and Garzarelli 2008). But as a project evolves and matures there are some tasks that are still in the exclusive domain of a few individuals. One example of this is the Project Leader, who is responsible for the coordination of production. Another example is package maintainers and core groups of programmers who usually have complete authority over their part of the project. This kind of situation is illustrated by individual *S* in Figure 3 who only oversees a specific set of components (represented by task 8) of the system by supervising the highly specialized team composed of s_1, s_2, s_3 and s_4. In another sense, the box within Figure 3 that encompasses s_1, s_2, s_3 and s_4 (under the guidance of *S*) depicts an instance of the cathedral mode within voluntary production. That is, we have a rudimentary depiction of conscious direction in a voluntary project.

Therefore, we have that certain programmers can be working solely on one task (horizontal division of labor) while others can be working on multiple tasks (similarly to the vertical division of labor[8]). But what are the implications of this? The mixture of vertical and horizontal divisions of labor in voluntary OSS organization implies that:

- any individual is able to self-select into a task that corresponds to his or her primary expertise and is at the same time also able to self-select into tasks that do not reflect the same match;
- any number of individuals possibly may be contributing to the same task at any given point in time.

The first implication suggests that the voluntary nature of OSS production yields a degree of imperfect matching of individual specialization to performed task.

[8] "Similarly" yet not identically to the vertical division of labor, because work – input-output relations – may be not necessarily linearly sequential, but rather roundabout.

The second implication highlights the possibility that, at any given moment, several individuals may be working spontaneously on the same production task. Prima facie, these two implications would lead us to classify voluntary OSS production as inefficient under standard production accounts. However, this is not necessarily the case once a more explicit focus is placed on the learning dynamics innate in voluntary OSS production: more knowledge is created when individuals can more freely interact with each other.

4. Mistake-ridden learning is not such a mistake

When a programmer is faced with some specific need, the voluntary nature of OSS production allows programmers the ability to self-select into tasks that ultimately service that need. But the correction of disequilibria in voluntary OSS production, as hinted, does not necessarily hinge on perfect matching between task and volunteer. As such, one might regard such imperfect sorting as having little or no productive value. There is nothing erroneous with this argument if one reasons according to familiar productivity measures, such as increases in productivity at lower unit costs. However, what matters are also overall increases in productivity tied to the growth of knowledge, namely, deriving from the learning curve (Garzarelli *et al.* 2008). The crux of the matter is that just as specialized contributions can be a source of productivity so too can non-specialized ones: as elaborated presently, voluntary OSS production promotes an environment that facilitates learning from the mistakes of one another.

In traditional division of labor accounts, such as the Smithian ones alluded to earlier, productivity gains from specialization originate from a cognitive process of *individual* learning-by-doing that is a "by-product" of production. For example, in

horizontal division of labor, repeating the same task and following the same routine means that through increasing specialization an individual learns to simplify his task. That is to say that productivity increases as a result of increased experience in production (Atkinson and Stiglitz 1969). As such, specializing individuals develop the knowledge to source innovative ways by which the production task may be altered so as to produce more efficiently.

But in such cases there is negligible learning among persons involved in the same production process: learning-by-doing is mostly individual rather than organizational. In voluntary OSS production, we saw, it is possible for many individuals to be working simultaneously on the same task at any given point in time. In this case there is scope for individuals involved in the same task to collaborate, to share their production failures and successes. As a result, learning-by-doing becomes social in that the collaborative network extends the boundaries of learning-by-doing beyond the individual level into the organizational one. Knowledge growth materializes through planned and unplanned organizational interactions that transmit and exchange knowledge. Organizational interaction helps an individual improve her problem solving ability by exposing her to the unique 'bits' of knowledge held by other individuals.[9] While the ability to work independently leads to improvements in one's skills.

Consider the following example. Suppose a problem is identified and that the problem corresponds to a programming task that neither programmer X nor Y are specialized in. If X and Y are motivated by some personal need to solve the problem, both programmers will choose to spontaneously contribute towards the task. Suppose also that programmer X is first to provide his modification of the source code to the

[9] Compare Marshall (1961, p. 271) on social learning: "if one man starts a new idea it is taken up by others and combined with suggestions of their own; and thus it becomes the source of further new ideas."

community of developers. If X's modifications are flawed, programmer Y will be able to identify where X went wrong, and thus avoid replicating the same mistake. Taking the new knowledge learnt from X's mistakes, Y can find an alternative solution to the problem, a solution that would have otherwise been unknown to Y had X's contribution not have been able to be effectively communicated. The mistake of one individual serves to pollinate the ideas of another in a manner that bears fruit to new, productive ideas. Each learning opportunity extends the innate knowledge of participants to the production process. This in turn expands the arsenal of knowledge that can be used to scrutinize any given problem at any given point in time. This example is admittedly crude, nonetheless it helps us to more sharply focus the dynamics of mistake-ridden learning found in voluntary OSS production. It is very likely that numerous individuals provide spontaneous contributions and that numerous other individuals identify and learn from different mistakes. With potentially thousands of individuals spontaneously contributing at any given moment, the potential 'cross-pollination' of ideas is vast.

The increase in learning experiences, in turn, favors an enhancement of learning capabilities, that is, of that specific production knowledge that is revealed in the reflexive process of adapting to changing circumstances.[10] In essence, by learning more, individuals learn how to learn better – they improve their learning skills. This "learning to learn" (Stiglitz 1989) involves improvements, firstly, in the capability to absorb information, and, secondly, in the capability to disseminate information, acknowledging mistakes and retaining best practices. In keeping with this view, we would expect heterogeneity in learning rates, that is, the open-ended sorting of

[10] On the general theoretical notion of capabilities in the context of organizational analysis, see for example Garzarelli (2008).

individual to task gives rise to slow and fast learners. And it is precisely this heterogeneity in learning rates that can be a vital source of knowledge gains.

"Organizations store knowledge in their procedures, norms, rules and forms" – their "code". "They accumulate such knowledge over time, learning from their members. At the same time, individuals in an organization are socialized to organizational beliefs." Thus, a mutual learning process exists whereby "the organizational code affects the beliefs of individuals, even while it is being affected by those beliefs" (March 1991, pp. 73 and 75). According to such perspective, knowledge gains transpire as a result of factors that create "variability" between organizational and individual beliefs. One way for achieving and sustaining this variability is for an organization to maintain a heterogeneous population of slow and fast learners. For "any average rate of learning from the code, it is better from the point of view of equilibrium knowledge to have that average reflect a mix of fast and slow learners rather than a homogeneous population" (March 1991, p. 77). This equilibrium entails that both the organization and a fraction of fast learning individuals are simultaneously able to learn from the "deviations" (i.e., the errors) of slow learners.[11] Consequently, being characterized by both specialized and non-specialized inputs, we would expect voluntary OSS to benefit from a mixture in learning rates in the same manner.[12]

Moreover, March continues, like the heterogeneity in learning rates, diversity among individuals' knowledge levels improves aggregate knowledge. The "old-timers" of an organization know more than the "new blood" of an organization. However, what they know is "redundant" with knowledge that is already possessed

---

[11] Equilibrium is established when all individuals and the organizational code share the same belief with respect to each dimension of reality. Thus, equilibrium entails the convergence of beliefs and the end of any variability.

[12] See David and Rullani (2008) for an empirical study germane to this claim; we came across this intriguing study only after the first draft of this paper was already completed.

and reflected by the organization. As a result, old timers are less likely to contribute new knowledge. The new blood introduced into the organization may be less knowledgeable than their existing counterparts, but their knowledge is less redundant with the existing state of organizational knowledge. Therefore, the entry of new individuals into an organization is more likely to contribute to new knowledge gains as long as collaboration continues to expose mistakes. One would imagine this to be particularly the case in voluntary OSS production where modularity renders the exit and entry of old timers and new blood relatively free.

The knowledge production from the open interaction of different knowledge stocks and flows moreover resonates with the main normative insight about modularity, namely, its superior resiliency to change (e.g., Simon 1998[1962]; Frenken *et al.* 1999). The ability of a modular system to continue working even if not all its parts are on the same page (pursuing an objective even if one module is malfunctioning, simultaneously trying to solve different problems, working at different rates, etc.) is what gives a modular system its edge.

Take software bugs, viz., those programming errors that could render the functioning of a software system less reliable. In the case of proprietary software, we saw, bug solving can seriously threaten the success of a project. This is so because given the unitary nature of the cathedral mode of production, a bug may halt the whole project for there may be a substantial lag between a bug's solution and the latter's adoption within the project. But this is not the case, as we also saw, for the bazaar. A bazaar can limp along even while several bugs are trying to be fixed. There is no need for all bugs to be necessarily fixed in order for the system to continue working, because, unlike the cathedral, information hiding allows the bazaar to work with a lower number of required communications among all parts of the system. The

voluntary division of labor of the bazaar in fact generates a distributed intelligence that is also, to some extent, parallel. That is to say that we have an organization where there is an uncommon level of redundancy as well as of uniqueness of knowledge content. It is the amalgamation of the redundancy and uniqueness – or, if you prefer, of non-specialized *and* specialized labor – from modularity that aids the speed of adaptation (e.g., bug fixing).[13]

In general, however, it would be imprecise to assume that adaptation – the creation and discovery of new knowledge – rests merely on a parallel and distributed intelligence. Here (as elsewhere) new knowledge can also be generated in isolation by mere thinking. "Any kind of experience – accidental impressions, observations, and even 'inner experience' not induced by stimuli received from the environment – may initiate cognitive processes leading to changes in a person's knowledge. Thus, new knowledge can be acquired without new information being received" (Machlup 1983, p. 644, emphasis removed).

But in these cases too it is the modular nature of voluntary OSS organization that plays a crucial role. Information hiding encourages individual abstract thinking: a volunteer can maintain congruency with the common goal of the project by working on a particular task that interests her, because she knows that while she is at work there will not be external disturbances. Similarly to property rights in social systems, in fact, information hiding defines sheltered domains where individuals can focus their cognitive attention not having to worry about possible 'violations' of property. The sheltering allows planning and acting notwithstanding the Hayekian knowledge problem, assisting divided labor (Miller and Drexel 1988).

---

[13] Organizational parallels with multi-level minds that are also modular have a long history in economics; see, for example, Marshall (1994[1867-8]), which seems to have had a profound influence on his subsequent theory of organization (Marshall, e.g., 1961, pp. 250-66).

However, with few exceptions mostly concerning novel knowledge simultaneously affecting several existing modules and their interrelations where we would see, for example, the coordination of the project leader, it is in the main the visible design rules that filter the value of the novel knowledge from individual thinking to the organization. Recall in fact that knowledge itself is a system of rules of action for a system that can also change a system. Hence, analogously to the case of organizationally-generated knowledge, it is the visible design rules – the already-accepted organizational knowledge – that most often coordinate the new knowledge that is individually generated. The visible design rules per se are a source of redundancy in that they are the minimum common denominator ordering the individual and organizational knowledge interactions, and that, additionally, allow knowledge to be re-used and shared across the system. In short, visible design rules cement a modular system while simultaneously being sufficiently plastic to allow changes in the system itself in the face of evolutionary necessity.

Let us point out before wrapping up that we are not suggesting that OSS holds a monopoly on voluntary production. One of the most obvious comparable organizational modes is arguably the production of science.[14] Scientific communities have organized themselves in a similar voluntary fashion at least since the late sixteenth century when inquiry replaced secrecy, social cooperation replaced individual isolation, and spontaneous coordination replaced top-down planned design and control (David, e.g., 2004). In addition, in most cases we decide what topics we fancy working on, and the decisions made need not be particularly founded on a 'best fit' for those topics as opposed to an interest in them. Similarly, we form research teams spontaneously for different projects, and we participate in networks and belong

[14] We are grateful to an anonymous referee who suggested that we make this parallel more explicit. Compare also the earlier Garzarelli *et al*. (2008) and Langlois and Garzarelli (2008) that hint to the production parallel with science as well as to others (e.g., voluntary production for hobbyist ends).

to 'invisible colleges' according to shared research topics and fields. To ensure the fit of what we do within the broader literature, we must posses, and need to convey, a sense of the broader structure of the scientific research program (note the word, program) within which we are operating. In this regard, new PhDs (the new blood) are often less clear about the broader architecture of their discipline than the more experienced researchers (the old timers). And just as open source development communities examine the robustness of submitted code, so too do peers of academia when refereeing papers submitted to journals and conferences. In both organizational modes, information is shared, suggestions for improvement made, learning opportunities created among colleagues, and contributions are more about spontaneously supplying effort to a big project rather than about making a product according to some predefined blueprint.[15]

5. Conclusion

This paper analyzes the modular organization of voluntary OSS production by considering the relationship between the imperfect matching of a programmer's specialization to a performed task and productivity. It proposes that productivity gains

[15] Two observations can be made. First, from casual empiricism, one could argue that it is more common for social scientists than for natural scientists to work in the bazaar mode. Second, voluntary production may not be safe from opportunism either (Williamson, e.g., 1985). A referee has an incentive to hold back on some things that she could see that she could develop in her own work, thereby allowing a paper to get published without some extensions being noted or 'bugs' fixed. In proprietary software firms, it is possible for an employee to make herself indispensable by limiting how much she documents what she does in an opportunistic manner, so that if she were to be fired it would take substantial investment for the organization to acquire the knowledge she had been using. We arguably get less potential for opportunism in voluntary OSS projects, though it is easy to imagine that if an OSS volunteer also has a 'day job' in which he uses similar capabilities, he may wish to limit how much he shares with others who are working on the same module and have day jobs with other companies who may be rivals. Note here that opportunism does not derive from physical asset specificity or other financial commitments that may be hard to liquidate or re-use elsewhere. Since exit in voluntary organization is easy, the wasted resource would merely be time, though this time investment may be more than offset by the learning that takes place while involved in the voluntary project. This second observation also brings to mind Richardson's (1960) concern with what he saw as the 'problem' of coordinating market entry decisions when entry barriers are low. In voluntary production, having 'too many' programmers working on a particular module or research project prima facie doesn't seem a particularly big problem because of ease of exit. Be that as it may, both observations require further scrutiny.

may also be realized through non-specialized inputs: imperfect matching is offset by a parallel and overlapping division of labor that aids learning from each other's mistakes. As long as volunteers can freely exchange the knowledge that they come across, generate, and interpret, organizational knowledge can only ever grow.

Voluntary production renders relatively easy for individuals to take up a set of tasks out of a sheer need to do so; however, it renders equally relatively easy for them to abandon a set of tasks in the pursuit another set. As a result, there may be a situation in which volunteers do not bring to completion their initial tasks. Therefore, voluntary production may not always be able to optimally capitalize on spontaneous contributions; and such inconsistency of efforts may also mean that development speed is not always as rapid as we would like to believe.

References

Ames, Edward, and Nathan Rosenberg 1965. "The Progressive Division and Specialization of Industries," *Journal of Development Studies* 1(4): 363-383(July).

Atkinson, Anthony B., and Joseph E. Stiglitz 1969. "A New View of Technological Change," *Economic Journal* 79(315): 573-578(September).

Baetjer, Howard Jr. 1998. *Software as Capital. An Economic Perspective on Software Engineering*. Los Alamitos, CA: IEEE Computer Society.

Baldwin, Carliss Y. 2008. "Where Do Transactions Come From? Modularity, Transactions, and the Boundaries of Firms," *Industrial and Corporate Change* 17(1): 155-195.

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

David, Paul A. 2004. "Understanding the Emergence of 'Open Science' Institutions: Functionalist Economics in Historical Context," *Industrial and Corporate Change* 13(4): 571-589.

David, Paul A., and Francesco Rullani 2008. "Dynamics of Innovation in an 'Open Source' Collaboration Environment: Lurking, Laboring, and Launching FLOSS Projects on SourceForge," *Industrial and Corporate Change* 17(4): 647-710.

Deci, Edward L. 1971. "Effects of Externally Mediated Rewards on Intrinsic Motivation," *Journal of Personal and Social Psychology* 18(1): 105-115.

Dosi, Giovanni, and Marco Grazzi 2006. "Technologies as Problem-solving Procedures and Technologies as Input-Output Relations: Some Perspectives on the Theory of Production," *Industrial and Corporate Change* 15(1): 173-202.

Feller, Joseph, and Brian Fitzgerald 2002. *Understanding Open Source Software Development.* London: Addison-Wesley.

Fransman, Martin F. 1994. "Information, Knowledge Vision and Theories of the Firm," *Industrial and Corporate Change* 3(3): 713-757.

Frenken, Koen, Luigi Marengo, and Marco Valente 1999. "Interdependencies, Near-decomposability and Adaptation," in Thomas Brenner, ed., *Computational Techniques for Modeling Learning in Economics*. Dordrecth: Kluwer: 145-165.

Frey, Bruno S. 1997. "On the Relationship between Intrinsic and Extrinsic Work Motivation," *International Journal of Industrial Organization* 15(4): 427-439(July).

Garzarelli, Giampaolo 2008. "The Organizational Approach of Capability Theory," *Review of Political Economy* 20(3): 443-453(July).

Garzarelli, Giampaolo, Yasmina Reem Limam, and Bjørn Thomassen 2008. "Open Source Software and Economic Growth: A Classical Division of Labor Perspective," *Information Technology for Development* 14(2): 116-135(Spring).

Georgescu-Roegen, Nicholas 1970. "The Economics of Production," *American Economic Review* 60(2): 1-9(May).

Giuri, Paola, Matteo Ploner, Francesco Rullani, and Salvatore Torrisi 2010. "Skills, Division of Labor and Performance in Collective Inventions. Evidence from Open Source Software," *International Journal of Industrial Organization* 28(1): 54-68(January).

Hayek, Friedrich A. von 1937. "Economics and Knowledge," *Economica* 4(13): 33-54(February).

Hayek, Friedrich A. von 1945. "The Use of Knowledge in Society," *American Economic Review* 35(4): 519-530(September).

Hayek, Friedrich A. von 1978. *New Studies in Philosophy, Politics, Economics and the History of Ideas*. London: Routledge & Kegan Paul.

Houthakker, Hendrik S. 1956. "Economics and Biology: Specialization and Speciation," *Kyklos* 9(2): 181-189.

Jensen, Michael C., and William H. Meckling 1992. "Specific and General Knowledge, and Organizational Structure," in W. Lars and H. Wijkander (eds.), *Contract Economics*. Oxford: Basil Blackwell: 251-274.

Knight, Frank H. 1967. *The Economic Organization* (*with an article Notes on Cost and Utility*). New York: Augustus M. Kelley Publishers (Reprints of Economic Classics). First edition privately printed in 1933; first published 1951.

Kogut, Bruce, and Anca Metiu 2001. "Open Source Software Development and Distributed Innovation," *Oxford Review of Economic Policy* 17(2): 248-264.

Langlois, Richard N., and Pierre Garrouste 1997. "Cognition, Redundancy, and Learning in Organizations," *Economics of Innovation and New Technology* 4(4): 287-299.

Langlois, Richard N., and Giampaolo Garzarelli 2008. "Of Hackers and Hairdressers: Modularity and the Organizational Economics of Open-source Collaboration," *Industry and Innovation* 15(2): 125-143(April).

Leijonhufvud, Axel 1986. "Capitalism and the Factory System," in Richard N. Langlois, ed., *Economics as a Process: Essays in the New Institutional Economics*. New York: Cambridge University Press: 203-223.

Leijonhufvud, Axel 2007. "The Individual, the Market, and the Division of Labor in Society," *Capitalism and Society* 2(2): Article 3.

Machlup, Fritz 1983. "Semantic Quirks in the Theory of Information," in F. Machlup and U. Mansfield (eds.), *The Study of Information: Interdisciplinary Messages*, New York, John Wiley: 641-671.

March, James G. 1991. "Exploration and Exploitation in Organizational Learning," *Organization Science* 2(1): 71-87(February).

Marshall, Alfred 1994[1867-8]. "Ye Machine," reprinted, as edited by Tiziano Raffaelli, in *Research in the History of Economic Thought and Methodology, Archival Supplement* 4: 116-32.

Marshall, Alfred 1961. *Principles of Economics* (ninth [variorum] edition, with annotations by C. W. Guillebaud), Volume I. London: Macmillan.

Miller, Mark S. and K. Eric Drexler 1988. "Markets and Computation: Agoric Open Systems," in Huberman, B. A. (ed.), *The Ecology of Computation*. Amsterdam: North-Holland: 133-76. Online version (last accessed May 1, 2010): http://e-drexler.com/d/09/00/AgoricsPapers/agoricpapers/aos/aos.0.html.

Mises, Ludwig von 1960. *Epistemological Problems of Economics*, Princeton, New Jersey, Toronto, New York and London: D. Van Nostrand Company, Inc. Translated by George Reisman. Originally published 1933.

Murdock, Ian 2003. "Debian: A Brief Retrospective," (last accessed April 26, 2010) http://www.linuxplanet.com/linuxplanet/editorials/4959/1/.

Osterloh, Margit, and Rota, Sandra G. 2004. "Open-source Software Development – Just Another Case of Collective Invention?" Working Paper, University of Zurich (March). Available at: http://ssrn.com/abstract=561744.

Parnas, David Lorge 1972. "On the Criteria to be Used in Decomposing Systems into Modules," *Communications of the ACM* 15(12): 1053-1058(December).

Raymond, Eric S. 2001. *The Cathedral and the Bazaar. Musings on Linux and Open Source by an Accidental Revolutionary* (revised edition). Sebastopol, CA: O'Reilly & Associates, Inc.

Richardson, G. B. 1960. *Information and Investment. A Study in the Working of the Competitive Economy*. London: Oxford University Press.

Rosenberg, Nathan 1969. "The Direction of Technological Change: Inducement Mechanisms and Focusing Devices," *Economic Development and Cultural Change* 18(1): 1-24(Part 1, October).

Simon, Herbert A. 1998. "The Architecture of Complexity: Hierarchic Systems," in *Idem*, *The Sciences of the Artificial*, 3rd edition, second printing. Cambridge, Mass.: MIT Press: 183-216. Originally published in 1962, *Proceedings of the American Philosophical Society* 106(6): 467-82(December).

Smith, Adam 1981. *An Inquiry into the Nature and Causes of the Wealth of Nations* (two volumes). Indianapolis: Liberty Fund. First published 1776.

Stiglitz, Joseph E. 1987. "Learning to Learn, Localized Learning and Technological Progress," in Partha Dasgupta and Paul Stoneman, eds., *Economic Policy and Technological Performance.* Cambridge: Cambridge University Press: 125-153.

Williamson, Oliver E. 1985. *The Economic Institutions of Capitalism.* New York: The Free Press.

Winter, Sidney G. 2005. "Toward an Evolutionary Theory of Production," in Kurt Dopfer, ed., *The Evolutionary Foundations of Economics*. Cambridge: Cambridge University Press: 223-254.