Of Hackers and Hairdressers: Modularity and the Organizational Economics of Open-source Collaboration

Richard N. Langlois
The University of Connecticut
U63 Storrs, CT 06269-1063 USA
(860) 486-3472 (phone)
Richard.Langlois@UConn.edu

Giampaolo Garzarelli
Università degli Studi di Roma, "La Sapienza"
Dipartimento di Teoria Economica
e Metodi Quantitativi per le Scelte Politiche
Piazzale Aldo Moro, 5
00185, Rome, Italy
+39.06.49910627 (phone)
ggarzarelli@tiscali.it

Draft: April 15, 2005

Special thanks to Roberto Galoppini and Martin Michlmayr.

Introduction.

We tend to think of decentralized intellectual collaboration as a phenomenon of open-source software, the Internet, and the New Economy.[1] But consider the plight of a certain Monsieur Prony, as recounted by Charles Babbage (1835, §243). Gaspard Riche de Prony (1755-1839) was a French civil engineer whom the Revolutionary government of 1790 had charged with an unenviable assignment: construct the largest and most accurate set of trigonometric and logarithmic tables ever produced. One day while pondering this seemingly impossible task, Prony wandered into a bookseller's shop and absent-mindedly thumbed through a copy of Adam Smith's *Wealth of Nations*. Suddenly it struck him. He could enlist the division of labor to construct the tables – in effect, he could manufacture logarithms like pins. Driven by this insight, Prony set up a collaborative project along the following lines. He would begin by enlisting four or five of the most eminent mathematicians in France to devise formulas well suited for numerical calculation. The results would then pass to a small team of run-of-the-mill mathematicians who would turn the formulas into simple algorithms. The actual calculations would be performed by a large team of 60 to 80, most of whom knew no math beyond simple addition and subtraction. Indeed, in the event the calculators came largely from the ranks of unemployed hairdressers, a group who had lost their once elaborately coiffed clients to the

[1] In the context of software, our use of the term "open-source" follows von Hippel and von Krogh (2003). We adopt an even more expansive meaning below.

same Revolutionary taste for austerity and reason that had inspired Prony's commission. "By 1794, 700 results were being produced each day" (Grattan-Guinness 1990, p. 180).

Obviously, this example differs from present-day open-source software projects in a number of respects. For one thing, Prony's calculators were not volunteers, and were presumably paid either as employees or on a per-calculation basis.[2] Moreover, the entire process was "fordist" (to use an anachronistic term) rather than collegial: the design was top down, with no communication among the calculators or feedback from them to the mathematicians above. And the calculators were not creative programmers but deskilled automatons whom Babbage had every hope of replacing with his planned difference engine.[3] Nonetheless, like present-day open-source efforts, the Prony Project was an attempt to share-out a complex creative task. Both are examples of what Babbage called the "mental division of labor."

This paper seeks to understand the phenomenon of open-source collaboration by placing it within this larger context of the intellectual division of labor. Our primary focus here will not be on the problem of incentives to

[2] Unfortunately, no information seems to have survived about the actual organization of the work (Grattan-Guinness 1990, p. 179).

[3] There is at least one recent analogue to the Prony Project: NASA's Clickworkers study, which enlisted volunteers to engage in well-specified and relatively menial astrometric tasks. See: http://clickworkers.arc.nasa.gov/top. Of course, Babbage's dream of handing off the most menial calculations to computers has long since come true, and today Google will help you donate free time on your Internet-connected computer to solving a small piece of a large scientific problem. See: http://toolbar.google.com/dc/offerdc.html.

contribute to such efforts, though we may be led back to the issue of incentives indirectly. Much of the existing literature, especially that created by economists (Lerner and Tirole 2002), has seen incentives as central: why would so many people contribute time and effort without pecuniary compensation or the promise of returns from intellectual property?[4] The answer seems to be that sources of motivation are abundant and adequate. We concentrate instead on what we take to be the more interesting issue: how does open-source collaboration *coordinate* the mental division of labor? In Prony's case, the answer was top-down design and control. But the hallmark of many present-day open-source efforts is bottom-up coordination. The Prony Project was a fordist cathedral; but the paradigmatic open-source software project of today is a spontaneous bazaar (Raymond 2001). Under which circumstances, and in what quantities, is central design necessary? When can genuinely decentralized design lead to a well-ordered and effective structure?

In effect, this paper seeks to ask for open-source collaborations the question Coase (1937) originally asked about firms: why do they exist? Although "open-source" has its original technical meanings in software design and law, we will appropriate the term here to refer to the general phenomenon of a spontaneously coordinated mental division of labor. Examples include the phenomenon of "collective invention" (Osteloh and Rota 2004) within industrial

[4] One exception is Benkler (2002), to which we return.

communities (Allen 1983) and the professions (von Hippel 1989; Savage 1994); the business of journal editing and refereeing familiar to academics (Bergstrom 2001); online open bibliographic databases, such as Research Papers in Economics (**RePEc**) (Krichel and Zimmermann 2005); cases of "open-source" collaboration for literary and hobbyist ends rather than for software production (like the online encyclopedia **Wikipedia** and the photography site **photo.net**); and even the modern practice of "open science" (David 1998).

We redirect the Coasean question toward open-source collaboration by adding an additional dimension to Coase's spectrum between market and firm. In Coase and the Post-Coasean literature, markets are about the exchange of products or outputs; such exchange is coordinated spontaneously, in the sense that relative prices rather than fiat direct resources. A firm stands in contrast to both of these aspects of markets: it replaces contracts for products with employment contracts, effectively substituting a factor market for a product market (Cheung 1983); at the same time, it replaces spontaneous coordination with some kind of central design or direction. Notice that this leaves two unexamined alternatives: product markets governed by central direction and factor markets coordinated spontaneously. Inside contracting and outsourcing are examples of the former. Open-source collaboration is an example of the latter. That is, *open-source collaboration is an organizational form that permits the exchange of effort rather than the exchange of products, and it does so under a regime in* *which suppliers of effort self-identify like suppliers of products in a market rather than accepting assignment like employees in a firm.*

Like Coase and the present-day economics of organization, we attempt to answer the "why" question by isolating the economic circumstances that favor one form of organization over another. As we have already hinted, however, we do not think that the lens of incentives, which focuses most of the present-day economics of organization, is the appropriate glass through which to see the phenomenon of open-source collaboration. We turn instead to what we may loosely call the modularity theory of the firm (Langlois 2002; Baldwin and Clark 2003), since, as we will argue, open-source collaboration ultimately relies on the institutions of modularity. Here the issue hinges on a tradeoff between the benefits (and costs) of modularity and the benefits (and costs) of its opposite, integrality. This tradeoff determines the extent to which central coordination is preferable to spontaneous or decentralized coordination. In this respect we are attempting to make more precise an idea that is already implicit in many discussions of the open-source phenomenon. In addition, however, we suggest that the nature and intensity of demand – especially the extent to which demand is quality or time sensitive – can shift the margin between modularity and integrality. Moreover, we argue, the existence of this tradeoff between modularity and integrality can constitute an incentive and "focal point" for technological or organizational change (Rosenberg 1976) – change that sometimes, and perhaps typically, shifts the margin in favor of modularity.

The economics of modularity and integrality.

In his famous account of the development of the operating system for the IBM 360 series of computers, Frederick Brooks (1975) paints a depressing portrait of the intellectual division of labor. Like Prony, Brooks parceled out the tasks of mental labor to a large number of workers. Unlike Prony, however, Brooks found himself faced with a daunting problem of coordination. An operating system is an immensely complicated tangle of interconnections, and every piece potentially depends on every other piece. Brooks took this to mean that every programmer ought to know what every other programmer is doing.[5] Unfortunately, however, since the number of connections in a network rises exponentially with the number of nodes, so must the costs of coordination rise exponentially as more workers are added to a project.[6] The implication, he reasoned, is that coordination costs quickly vitiate any benefits from the intellectual division of labor.

At about the same time, however, David Parnas (1972) and other researchers were looking at software systems in a different light. Solving the problem of interdependency, they argued, is not a matter of maximizing communication among the parts but rather of *minimizing* communication. In this approach, which laid the foundations for object-oriented programming, one

[5] As open-source guru Eric Raymond observes, Brooks's dire conclusion rested on the implicit "assumption that the communication structure of [a] project is necessarily a complete graph" (Raymond 2001, p. 35).

attempts to design systems in which not only do the parts not need to communicate richly with one another but are actually *forbidden* from communicating with one another. The basic idea is that "system details that are likely to change independently should be the secrets of separate modules; the only assumptions that should appear in the interfaces between modules are those that are considered unlikely to change" (Parnas *et al*. 1985, p. 260). This is the notion of *information hiding* or *encapsulation*.

Software design reveals the logic of modularity with particular clarity. But the basic ideas were long ago articulated by Herbert Simon (1962) in a more general context. Simon contrasted systems that are *non-decomposable* with systems that are (or are nearly) *decomposable*. Brooks's conception of the 360 operating system created a non-decomposable system: every part communicates with virtually every other part.[8] By contrast, a decomposable system is one in which all (or most) communication takes place *within* the subsystems and little or none *among* the subsystems. As Simon put it more recently, a decomposable system "can be thought of as a boxes-within-boxes hierarchy with an arbitrary number of levels. Its special characteristic is that equilibrating interactions

6 Brooks (1975) actually suggested that costs should rise as the *square* of the number of workers, an idea now sometimes called Brooks's Law.

7 To put it in a different technical language, Brooks's dire conclusion rested on the implicit "assumption that the communication structure of [a] project is necessarily a complete graph" (Raymond 2001, p. 35).

8 As Baldwin and Clark (2003) rightly insist, the notion of "communication" should involve more than information, and can include the transmission of energy and materials as well as symbols.

between boxes at any level take place much more rapidly than the interactions between boxes at that same level, and similarly all the way to the top of the hierarchy." (Simon 2002, p. 589).

Consider Figure 1, where an entry of x in location a_{ij} means that element a_i communicates with element a_j. Matrix 1 is a fully non-decomposable system: every element communicates with every other element. By contrast, Matrix 2 is a relatively decomposable system. Communication is encapsulated within clusters of elements that do not communicate with elements "far away." Notice, however, that, although a nearly decomposable system like Matrix 2 clearly solves the problem of coordination, it does so by throwing the baby out with the bath water. Matrix 2 is a congeries of autarkic clusters that

$$
\begin{array}{c|ccccccc|}
 & a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 \\
\hline
a_1 & x & x & x & x & x & x & x \\
a_2 & x & x & x & x & x & x & x \\
a_3 & x & x & x & x & x & x & x \\
a_4 & x & x & x & x & x & x & x \\
a_5 & x & x & x & x & x & x & x \\
a_6 & x & x & x & x & x & x & x \\
a_7 & x & x & x & x & x & x & x \\
\hline
\end{array}
$$

1. A non-decomposable system.

$$
\begin{array}{c|ccccccc|}
 & a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 \\
\hline
a_1 & x & x & & & & & \\
a_2 & x & x & & & & & \\
a_3 & & & x & x & & & \\
a_4 & & & x & x & & & \\
a_5 & & & & & x & x & \\
a_6 & & & & & x & x & \\
a_7 & & & & & & & x \\
\hline
\end{array}
$$

2. A nearly decomposable system

$$
\begin{array}{c|ccccccc|}
 & a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 \\
\hline
a_1 & x & x & x & x & x & x & x \\
a_2 & x & x & x & & & & \\
a_3 & x & x & x & & & & \\
a_4 & x & & & x & x & & \\
a_5 & x & & & x & x & & \\
a_6 & x & & & & & x & x \\
a_7 & x & & & & & x & x \\
\hline
\end{array}
$$

3. A modular system with common interface.

Figure 1.

eliminates the costs of cooperation by the expedient of eliminating cooperation.

The term *modular system* takes on many meanings in the literature; but one important candidate definition, which we adopt here, is that a modular system is a nearly decomposable system that preserves the possibility of cooperation by adopting a common interface. The common interface enables, but also governs and disciplines, the communication among subsystems.[9] In terms of Figure 1, an interface would be a set of elements that communicates with most or all the other elements. In Matrix 3, element a_1 is the common interface: a_1 communicates with all the a_{ij} and all the a_{ij} communicate with a_1.[10] In other respects, however, Matrix 3 remains sparse off the diagonal.

Let us refer to a common interface as *lean* if it enables communication among the subsystems without creating a non-decomposable system, that is, if it enables communication without filling up the off-diagonal. As we will see, an interface may become standardized; it may also be "open" as against "closed." But it is the leanness of the interface, not its standardization or openness, that makes a system modular. Baldwin and Clark (2000) suggest thinking about modularity in terms of a partitioning of information into *visible design rules* and *hidden design parameters*. The visible design rules (or visible information) consist

9 This depiction of an interface is only meant to be suggestive. For one thing, cooperation may not require that communication be completely bidirectional in all cases, that is, it may not require that row 1 and column 1 be fully populated. Moreover, many different kinds of interface configurations are possible. On this point see Ulrich (1995).

10 Think of a_1 as M. Prony's assistant, who must have had to scurry among the calculators, each one working in isolation, to gather up the results and arrange them for typesetting.

of three parts. (1) An *architecture* specifies what modules will be part of the system and what their functions will be. (2) *Interfaces* describe in detail how the modules will interact, including how they fit together and communicate. And (3) *standards* test a module's conformity to design rules and measure the module's performance relative to other modules. Notice that "standards" in this sense doesn't necessarily imply that the architecture or interfaces are standard in the sense of being publicly shared or common to many similar artifacts or systems. A personal computer, for example, could be modular in Baldwin-and-Clark's sense but still be a unique design incompatible with the architecture and interfaces of other computers.[11]

All in all, then, modularity seems a powerful and elegant solution to the problem of coordinating the intellectual division of labor.[12] No less a figure than Linus Torvalds has testified to the value of modularity in the arena of open-source software.

> With the Linux kernel it became clear very quickly that we want to have a system [that] is as modular as possible. The [open-source] development model really requires this, because otherwise you can't easily have people working in parallel. It's too painful when you have people working on the same part of the kernel and they clash.
>
> Without modularity I would have to check every file that changed, which would be a lot, to make sure nothing was changed that would effect anything else. With modularity, when someone sends

[11] To put it another way, the term "standards" is sometimes used to mean a set of normative criteria and sometimes to imply replication. Although ultimately related, these are two quite different things. On the various meanings of the term, see David (1987).

[12] Garud *et al*. (2003) collects together many of the foundational articles on modular systems.

> me patches to do a new filesystem and I don't necessarily trust the patches *per se*, I can still trust the fact that if nobody's using this filesystem, it's not going to impact anything else. … The key is to keep people from stepping on each other's toes. (Torvalds 1999, p. 108.)

Nonetheless, we argue, there can be costs to modularity as well as benefits, and there can be benefits to non-decomposability – or *integrality*, as we will call it – as well as costs. Indeed, the two are opposite sides of the coin. What modularity does well integrality does poorly, and what integrality does well modularity does poorly. The costs of the one are essentially the foregone benefits of the other.

The first kind of benefit from modularity has already occupied us extensively: the ability of a modular system to obviate widespread communication among the modules (or their creators) and to limit unpredictable interactions. In effect, the process of modularization unburdens the system's elements of the task of coordination by handing that function off to the visible design rules. Coordination is imbedded or institutionalized in the structure of the system, which means that it doesn't have to be manufactured on the spot by the participants.

A second source of benefits derives from what Garud and Kumaraswamy (1995) call *economies of substitution*.[13] To the extent that the interfaces governing

[13] We can think of these economies of substitution as a species of what economists call economies of scope. Economies of scope exist when it is cheaper to make a given product if you are already making similar products than if you were to start from scratch. This is possible to the extent that you can reuse existing fixed investments (including knowledge)

the modular system are sufficiently standardized, it may be possible to upgrade a system by piecemeal substitution of improved modules without having to redesign the entire system. In large open-source software projects, for instance, the "approach of substituting individual components is the norm."[14] It may also be possible to optimize a system by choosing the best available modules or to customize a system to one's tastes or needs by selecting only some modules and not others[15] (Langlois and Robertson 1992).

A third, and perhaps most important, benefit of modularity is that it militates in favor of specialization and the effective use of local knowledge. If we do not subdivide tasks, everyone must do everything, which means that everyone must know how to do everything. But, as Babbage understood, if we do subdivide tasks, we can assign workers according to comparative advantage. Why pay a mathematician for those parts of the work that an (unemployed) hairdresser could do? More significantly, however, a modular system can do more than use a given allocation of local knowledge effectively – it can potentially tap into a vast supply of local knowledge (Langlois and Robertson

---

instead of reinventing the wheel. Economies of scope are normally discussed as a property of the "production function" of a firm. But as Langlois and Robertson (1995, p. 5) argue, there can be *external* economies of scope in an open modular system, since the visible design rules constitute a shared fixed investment that everyone can reuse in creating products through substituting and recombining modules. On external scope economies in the case of software, see Baetjer (1998, p. 99).

14 Martin Michlmayr, Debian GNU/Linux Project Leader until April 17, 2005, personal communication, 31 December 2004.

15 Bessen (2001) and Kuan (2001) see the benefits of open-source software production in terms of its ability to fine tune the product to user needs by making users part of the production process.

1992). This has not been lost on open-source developers, who often wax poetic on the ability of the open-source model to tap into a larger "collective intelligence."[16] Raymond (2001) even installs a version of this idea as "Linus's law": "Given enough eyeballs, all bugs are shallow." That is to say: "Given a large enough beta-tester and co-developer base, almost every problem will be characterized quickly and the fix obvious to someone" (Raymond 2001). A modular system increases the potential number of collaborators; and the larger the number of collaborators working independently, the more the system benefits from rapid trial-and-error learning[17] (Nelson and Winter 1977; Langlois and Robertson 1992; Baldwin and Clark 2000).

Notice here that, unlike the first two kinds of benefits from modularity – institutionalized coordination and economies of substitution – the benefits of tapping into "collective intelligence" depend not only on the technological characteristics of the system itself but also on the way the intellectual division of labor is organized. M. Prony could have set his hairdressers to work under one roof or he could have "put out" the calculations to them in their homes; he could

[16] For example: "The power of the commercial open-source business model is that you're tapping into the collective intelligence of your community." (John Roberts of SugarCRM, quote in LaMonica (2004).)

[17] As Raymond further elaborates, "while coding remains an essentially solitary activity, the really great hacks come from harnessing the attention and brainpower of entire communities. The developer who uses only his or her own brain in a closed project is going to fall behind the developer who knows how to create an open, evolutionary context in which feedback exploring the design space, code contributions, bug-spotting, and other improvements come from hundreds (perhaps thousands) of people" (Raymond 2001, pp. 50-1).

have paid them by the calculation or by the hour. Presumably one set of alternatives would have worked best, and the economics of organization might shed light on which set that would be. But in all cases, the modularization into simple calculations obviated communication among the calculators and with their handlers. Similarly, IBM could have – and probably did – benefit from economies of substitution in the hardware aspects of the 360 series of computers, even though the system was emphatically closed to outside collaborators, the so-called plug-compatible vendors.[18] But for a modular system to take advantage of extended localized knowledge, the organization of the intellectual division of labor is no longer immaterial. In order to tap into "collective knowledge," the system's interface must be not only lean but also standardized and open.[19]

The economics of networks has taught us that, despite the occasional subtlety, standardization is the easy part. If interfaces are sufficiently lean and sufficiently open, there is a tendency for one of them to emerge as a dominant standard (Shapiro and Varian 1998). Openness is the more interesting issue. As the community of open-source software developers clearly understands, openness does not mean only unfettered access to *knowledge* of the visible design rules of the system, though that may be a necessary condition. Rather, openness

18 Who famously sued IBM under antitrust laws in an attempt to open access to the system. *United States v. International Business Machines Corporation*, 1956 Trade Cas. #68, 245 (S.D.N.Y. 1956).

19 These are not absolute concepts, of course. One can talk about degrees, and also kinds, of openness. So too with standardization.

is about the *right* to take advantage of those design rules.[20] More broadly, the degree of openness of a modular system is bound up with the overall assignment of *decision rights* within the intellectual division of labor.[21] This is an organizational issue and – what may be the same thing – a *constitutional* issue.

As Jensen and Meckling (1992, p. 251) point out, economic organization must solve two different kinds of problems: "the rights assignment problem (determining who should exercise a decision right), and the control or agency problem (how to ensure that self-interested decision agents exercise their rights in a way that contributes to the organizational objective)."[22] All other things equal, efficiency demands that the appropriate knowledge find its way into the hands of those making decisions. There are basically two ways to ensure such a "collocation" of knowledge and decision-making: "One is by moving the knowledge to those with the decision rights; the other is by moving the decision

20 The question of rights in open-source software is a much-discussed issue, revolving around the various kinds of software licenses possible or in actual use. It would take us too far afield to enter this complex area. But for some distinctions see Wheeler (2003, n.d.1, n.d.2).

21 At this point we have in mind the allocation of decision rights within the intellectual division of labor – for example, the allocation of decision rights to programmers in the process of designing a piece of software. But the same logic applies at the level of the system – *e. g.*, the software – itself. In that case, we can say that the visible design rules themselves are a kind of "constitution" that determines the rights of action that the elements of the system enjoy. And here there is a strong analogy between the idea of encapsulation in software design and private property rights in legal systems (Miller and Drexler 1988; Langlois 2002). In both cases, the operative principle is the creation of a protected sphere of activity permitting autonomy of action. The two levels overlap to the extent that a particular encapsulated subsystem is the property of a particular individual or group within the intellectual division of labor.

22 It is in this guise – the agency problem – that incentives have reappeared in our story. This is not exactly the same issue as the incentive to contribute, though it may ultimately be related.

rights to those with the knowledge" (Jensen and Meckling 1992, p. 253). These two choices – as well as possible variants and hybrids – are "constitutions" that set out the assignment of decision rights. Such assignments can take place within firms or within wider networks of independent collaborators.[23] In what follows, we distinguish three models: *corporate, hybrid*, and *spontaneous*.

In the case of the Prony Project, and of fordist production generally, decision rights remain centralized. This is because there is very little knowledge that needs to be transmitted; tasks have been made exceedingly simple, and the important knowledge – that involving design – is already at the center. The agency problem can be addressed either through investments in monitoring or by aligning incentives using a suitable piece rate. Even when the subdivided tasks are far more complicated – and require far more skill and creativity – it is still possible to organize the intellectual division of labor in more-or-less the same way, what we will call the *corporate* model. In this model, the ultimate decision rights remain centralized, even as many de facto decision rights are parceled out to employees at various levels of the hierarchy. Clearly, such an arrangement complicates the agency problem, since keeping everyone on the same page is no longer a simple matter of monitoring or incentive alignment in a narrow pecuniary sense.

[23] On the idea that firms are constitutional systems, see Gifford (1991).

Many would argue that, even within the corporate context, effective management of high-human-capital projects requires recourse to more "participatory" or collaborative models (Minkler 1993). Does this mean that there is really no difference between the corporate model and the hybrid or voluntary ones? The answer is no, for two reasons. First, as we have seen, even a large organization is bounded in the capabilities on which it can draw, and this limitation may be important in many cases. Second, the location of the ultimate decision right matters. For any division of intellectual labor we choose, behavior and performance will be different if we assign decision rights to some central authority rather than to the individual collaborators.[24]

The opposite of a corporate model would a fully decentralized one in which the collaborators retain the ultimate decision rights. But just as the central holder of decision rights in a corporation must in practice cede de facto decision rights to others, so in a decentralized system the collaborators must give up some pieces of their rights in order to collaborate. In a classic market narrowly understood, the collaborators do this through contract. In return for compensation from you, I choose to exercise my right to make shoes by

[24] Oliver Williamson (1985, p. 136) traces this effect to the phenomenon of "selective intervention," the tendency of the central holder of decision rights to meddle in the decisions of the collaborators. Henry Hansmann (1996) reminds us that those who possess de facto decision rights will be constrained in their exercise of those rights if they lack the ultimate or "formal" decision rights (which Hansmann equates with ownership). The modern corporation, he points out, is precisely an example in which it pays to assign the formal ownership rights to parties (the stockholders) who may actually be in a poor position to exercise effective control – in part because such an allocation keeps the rights out of the hands of other parties (notably the managers) who would abuse them.

producing the kelly green golf shoes you have contracted for.[25] In the limit, however, I may not even know who "you" are, and I exercise my decision rights in the direction of kelly green golf shoes in the hope that you, or someone like you, comes along. A classic market of this sort is an example of what we call the *spontaneous* model – spontaneous in the sense that the division of labor itself emerges (in the limit) from the choices of the collaborators rather than from a central designer.

This is a perspective on the firm/market dichotomy somewhat different from what one usually encounters in the economics of organization. More typically, one hears the following kind of story: markets are good at exchanging *products* for compensation, whereas firms are good at exchanging *effort* for compensation. The economics of organization can be understood from this perspective as a set of stories about why it is often costly to cooperate by trading products and often necessary to cooperate by trading effort.[26] Ever since Coase (1937), it has been more-or-less taken for granted that the only way to trade effort is through an employment contract: I pay for your time and the right to direct your effort within agreed limits (Simon 1951). In other words, the only way to trade effort is by setting up a firm. Perhaps the most intriguing aspect of the open-source model is that it flies in the face of this assumption: under the right

[25] Apologies to Deirdre McCloskey (1995).

[26] This is not a denial that effort is ultimately a product. It is merely a claim that effort is a particular kind of product, one whose properties make it costly to trade using only the relatively rudimentary institutional support that an anonymous spot market offers.

circumstances, it is possible to cooperate spontaneously on the effort margin, not just the product margin.[27] Rather than giving up their decision rights to others, open-source collaborators combine effort "voluntarily." Voluntarily here means not that the collaborators do not receive pecuniary compensation (though that may often be true) but rather that the collaborators choose their own tasks. Assignment of individuals to tasks – and, to an extent we will explore, even the overall design of the division of labor itself – arises from these voluntary choices, in much the same way that assignment of sellers to products in a classic market arises from self-selection.

[27] As we argued earlier, of course, this model actually antedates software. Notably, it has been the normal mode of organization in the professions (Savage 1994), including those that produce science (David 1998). But only with the prominence of open-source software development has the phenomenon begun to gain the attention of the economics of organization.

	Don’t self-identify	Self-identify
Products	Inside contracting Outsourcing	Classic market
Effort	Classic firm	Voluntary production

Figure 2

We can distinguish four possibilities. Consider Figure 2. Along one dimension is the issue of design: is assignment to task (and maybe even the division of labor itself) generated through the a centralized process or does it arise from the self-identification of collaborators with tasks? Along the other dimension is the problem of information and agency: are we talking about products cleanly measured and priced or are we taking about exchanges of effort that involve costs of measurement and agency? In the upper left-hand box, the division of labor is centrally designed, but the products of that labor are easily measured and priced. This is the world of inside or outside contracting. It is the Prony Project, as well as the outsourcing of intellectual activities like call centers, back-office work, or the reading of X-rays. In the lower left-hand box, the division of labor remains centrally designed, but the cost of measuring and pricing transactions makes it cheaper to purchase the effort of collaborators directly. This is the classic firm. In the upper right-hand box, participants self-select their contributions; but measurement and pricing costs are not prohibitive, and those contributions take the form of products offered on spec. This is the classic market. Finally, in the lower right-hand box, participants self-select their contributions (also effectively "on spec"); but those contributions come directly in the form of effort rather than of effort embodied in a product. This is the model of *voluntary* or open-source production.[28]

The most extreme form of a voluntary arrangement would occur when the self-selection of the collaborators itself actually creates the division of labor. This is far from unimaginable: it is exactly what happens in "the market" in the largest sense – including the market for software in the large. It also arguably happens in the context of academic open science, where the pattern of knowledge emerges from the self-selected research choices of the participants. If we cast our gaze down to a less lofty level, however, there almost always seems to be some pre-existing structure of possible tasks from which the participants choose. If Thomas Kuhn (1970) and others are right, even scientific researchers

[28] This two-dimensional schema has advantages, we argue, over the tripartite distinction Benkler (2002) offers among markets, hierarchies, and what he calls *peer production*. Benkler argues that a perceptible trend toward the increased importance of human capital in production is leading toward peer production and away from both markets and hierarchies. It may well be that, with economic growth and an expanding extent of the market, there is a general trend rightward in Figure 2, what Langlois (2003) calls the phenomenon of the Vanishing Hand. But an increased importance of human capital and greater spontaneity of production is consistent with markets as well as with decentralized collaboration through direct effort. Even apart from the likes of books, musical scores, or screenplays, there are a plethora of "consulting" services – from legal representation to brain surgery – that are priced on markets. The interesting issue, in the language of Baldwin and Clark (2003), is: when can cooperation be effectively measured and priced (and thus turned into "transactions") and when not?

are often – and maybe always – guided in their choice of problem by the constraints of earlier models and approaches. And at the level of any particular software project, the self-selection of workers to tasks takes place within the context of an established architecture or (at the very least) an established technological trajectory.[29]

Why is this so? Consider the experiments conducted by Kevan Davis, a British software engineer (Thompson 2004). Davis set up a website on which visitors could vote, pixel by pixel, on the design of a type font.[30] The results were quite presentable and, at least when cleaned up a bit, looked "like a mildly punk version of Helvetica, with occasional flashes of creative weirdness, such as the jaunty serif on the foot of the letter 'J'" (Thompson 2004). But when Davis asked people to draw a face or a television set by voting on pixels, the result was a shapeless mess. Efforts at drawing a goat looked mildly goat-like for a while, then collapsed into a jumble after 7,000 votes. The difference between a type font and a goat, of course, is that we come prepared with structural preconceptions that are far tighter in the former case than in the latter. For the font problem, there is something much closer to a pre-existing design architecture to guide individual contributions. This shouldn't be surprising in view of our discussion

[29] The idea of a technological trajectory is an application to technology of Kuhn's idea of a scientific paradigm (Dosi 1982).

[30] **http://www.typophile.com/smallerpicture/**

of modularity. Modularity enables large-scale cooperation; but it requires agreed-upon visible design rules.[31]

And here we finally arrive at the other side of the coin: the costs of modularity. The first of these is the (fixed) cost of establishing the visible design rules (Baldwin and Clark 2000). A (nearly) decomposable system may solve coordination problems in an elegant way, but designing such a system may take a considerable amount of time and effort.[32] There may also be costs to communicating the design rules to participants and securing agreement on them.

Another cost is that, at least in principle, it may not be possible to fine-tune a modular system as tightly as an integral one. For many kinds of software, this may no longer be an important issue in the face of Moore's law. But for other kinds of systems, there may be important performance losses from building a system out of modules. Automobiles, for example, may be have an inherent "integrality" that prevents automakers from taking advantage of modularity to the same degree as, say, makers of personal computers (Helper and MacDuffie 2002). One can't swap engines like one swaps hard drives, since a different engine can change the balance, stability, and handling of the entire vehicle. Clayton Christensen and his collaborators (Christensen, Verlinden, and Westerman 2002) have argued that integral designs, which can take advantage of

[31] Which may, as in the case of letters, be tacitly known design rules.

systemic fine-tuning, have an advantage whenever users demand higher performance than existing technology can satisfy. As the fine-tuned system continues to improve in performance, however, it will eventually surpass typical user needs. At that point, these authors argue, production costs move to the fore, and the integral system (and the integrated organization that designed it) will give way to a network of producers taking advantage of the benefits of modularity discussed earlier.[33]

A third, closely related, cost of modularity (benefit of integrality) is the tendency of modular systems to become "locked in" to a particular system decomposition. At least to the extent that knowledge gained creating one modularization of the system cannot be reused in generating a new decomposition, it is a relatively costly matter to engage in systemic change of a modular system, since each change requires the payment anew of the fixed cost of setting up visible design rules. If in addition an interface has become a standard, the problems of lock-in are compounded in the way popularized by Paul David (1985), since in that case many people would have simultaneously to

[32] Garud and Kumaraswamy (1995) cite evidence that the cost of designing a reusable modular software object may be as much as ten times the cost of designing software intended to be used only once.

[33] One might also add, however, that sometimes a modular system can improve in performance even faster than a fine-tuned system. To the extent that such a system benefits from "collective intelligence" and rapid-trial-and-error learning, the improvement in the parts can dominate any benefits from fine-tuning. Personal computers are again a case in point. PCs have come to outperform first mainframes, then minicomputers, then RISC workstations, all of which, in their day, made their money as fine-tuned non-modular systems (at least relative to PCs). Again, the extent to which modular innovation can outperform fine-tuning may depend on the degree of inherent integrality in the system.

pay the fixed cost of change. A modular system is good at modular or *autonomous* innovation, that is, innovation affecting the hidden design parameters of a given modularization but not affecting the visible design rules. But a modular system is bad at *systemic* innovation, that is, innovation that requires simultaneous change in the hidden design parameters and the visible design rules – simultaneous change in the parts and in the way the parts are hooked together.[34]

The benefits of an integral system in systemic change are related to the benefits of fine-tuning to which Christensen points. Fine-tuning is after all systemic change to improve performance. Thus integral systems may have advantages not only when users demand high performance in a technical sense but also when they need performance in the form of change and adaptability. This latter may also be a function of how *quickly* the user needs the system to perform; the front-end costs of a modular system may take the form of time costs – the output forgone while waiting for the modularization to crystallize or the visible design rules to get worked out. If a modularization is already in place, of course, the system can adapt and respond quickly by simply plugging in new

[34] The terms autonomous and systemic are from Teece (1986). There is a third possibility, what Henderson and Clark (1990) call *architectural* innovation. Here the modules remain intact, but innovation takes place in the way the modules are hooked together. (For a paradigmatic example of this kind of innovation, visit **Legoland**.) The possibility of architectural innovation underlies the benefits of economies of substitution discussed earlier.

modules to suit user needs.[35] But if there is not yet a modularization, or if the user needs a level of performance greater than can be achieved even with the best possible assortment of available modules, then an integral system may do better.

In terms of our earlier distinction between the corporate model and the spontaneous or voluntary model, the need for performance and rapid adaptability would tend to militate in the direction of the corporate (Langlois 1988). But this does not mean that unsatisfied needs for performance and rapid systemic adaptation therefore call for central planning on a Soviet scale. In Christiansen's account, unmet performance needs do always call for an integrated corporate structure. But the network theorist Duncan Watts (2004) reminds us that a decentralized structure, with its ability to utilize "collective intelligence," can sometimes be marshaled even in the service of an emergency response. His example is the way the Toyota Corporation responded in 1997 when the sole plant supplying a crucial component burned to the ground, threatening to bring production of an entire model to a halt. Rather than attempting to create centrally a new plant to make the component, Toyota instead tapped the knowledge and capabilities of a large number of its divisions and outside supplier with the intent of generating rapid trial-and-error learning. "More than 200 companies reorganized themselves and each other to develop at

[35] It is for this reason that Figure 3 grades modular systems as "B" rather than "C" on their ability to fine tune performance or adapt systemically. Modular systems can in fact fine tune

least six entirely different production processes, each using different tools, different engineering approaches, and different organizational arrangements. Virtually every aspect of the recovery effort had to be designed and executed on the fly, with engineers and managers sharing their successes and failures alike across departmental boundaries, and even between firms that in normal times would be direct competitors" (Watts 2004). Within a week, production of the component was back to pre-fire levels.

---

and adapt systemically to some degree by taking advantage of economies of substitution.

Clearly, of course, this response was not spontaneous in our sense. It was centrally directed and coordinated to a large extent. But neither was it the standard corporate model. Rather, it is an example of what we call a *hybrid*

	Modularity	Integrality
Communications costs	A	C
Economies of substitution	A	C
"Collective intelligence"	A	C
Set-up costs	C	A
Fine tuning	A	B
Systemic adaptation	A	B

Figure 3. Grading the alternatives.

model – one that has elements both of spontaneous, self-selected production and of central design. Such hybrid models are actually far more typical of open-source software development than are genuinely voluntary or spontaneous ones. Indeed, perhaps *all* models of open-source software development are hybrid models. In essence, a hybrid model is an attempt to get around the tradeoffs summarized in Figure 3. As Rosenberg (1976) has taught us, tradeoffs are a prime focal point for innovation. To the extent that innovative hybrid arrangements are able to inject some of the benefits of central design and integrality into a modular structure, such innovations will tend to make modular arrangements more valuable and more widespread.

References.

Allen Robert C. 1983. "Collective Invention," *Journal of Economic Behavior and Organization* **4**(1): 1-24 (March).

Babbage, Charles. 1835. *On the Economy of Machinery and Manufactures.* London: Charles Knight, fourth edition. Avaliable at: ‹**http://socserv2.socsci.mcmaster.ca/~econ/ugcm/3ll3/babbage/**›.

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

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

Baldwin, Carliss Y., and Kim B. Clark 2003. "Where Do Transactions Come from?" Working Paper, Harvard Business School, February. Available at: ‹**http://www.people.hbs.edu/cbaldwin/DR2/TransactionsFeb5v4.pdf**›.

Benkler, Yochai 2002. "Coase's Penguin, or, Linux and the Nature of the Firm," *Yale Law Journal* 112(3): 369-446 (December). Available at: ‹**http://www.yale.edu/yalelj/112/BenklerWEB.pdf**›.

Bergstrom, Theodore C. 2001. "Free Labor for Costly Journals?," *Journal of Economic Perspectives* **15**(3): 183–198 (Summer).

Bessen, James. 2001. "Open-source Software: Free Provision of Complex Public Goods." Available at: ‹**http://www.researchoninnovation.org/opensrc.pdf**›.

Brooks, Frederick P. 1975. *The Mythical Man-Month: Essays on Software Engineering*. Reading, MA: Addison-Wesley.

Cheung, Steven N. S. 1983. "The Contractual Nature of the Firm," *Journal of Law and Economics* **26**: 122 (April).

Christensen, Clayton M., Matt Verlinden, and George Westerman. 2002. "Disruption, Disintegration, and the Dissipation of Differentiability," *Industrial and Corporate Change* **11**(5): 955-993 (November).

Coase, Ronald H. 1937. "The Nature of the Firm," *Economica* (N.S.) **4**(16): 386-405 (November).

David, Paul A. 1985. "Clio and the Economics of QWERTY," *American Economic Review*, Papers and Proceedings **75**(2): 332-337 (May).

David, Paul A. 1987. "Some New Standards for the Economics of Standardization in the Information Age," in Partha Dasgupta and Paul Stoneman (eds.), *Economic Policy and Technological Performance.* Cambridge: Cambridge University Press: PAGES.

David, Paul A. 1998. "Common Agency Contracting and the Emergence of 'Open Science' Institutions," *American Economic Review* **88**(2): 15-21(May).

Dosi, Giovanni. 1982. "Technological Paradigms and Technological Trajectories," *Research Policy* **11**(3): 147-162 (June).

Garud, Raghu, and Arun Kumaraswamy. 1995. "Technological and Organizational Designs for Realizing Economies of Substitution," *Strategic Management Journal* **16**: 93-109 (Summer; Special Issue: Technological Transformation and the New Competitive Landscape).

Garud, Raghu, Arun Kumaraswamy, and Richard N. Langlois, eds. 2003. *Managing in the Modular Age: Architectures, Networks and Organizations*. Oxford: Blackwell Publishing.

Garzarelli, Giampaolo. 2004. "Open-source Software and the Economics of Organization," in J. Birner and P. Garrouste (eds.), *Markets, Information and Communication.* London and New York: Routledge: 47-62.

Grattan-Guinness, Ivor. 1990. "Work for the Hairdressers: The Production of de Prony's Logarithmic and Trigonometric Tables," *IEEE Annals of the History of Computing* **12**(3): 177-185 (July-September).

Hayek, Friedrich A. 1948. *Individualism and Economic Order.* Chicago: Chicago University Press.

Helper, Susan, and John Paul MacDuffie. 2002. "B2B and Modes of Exchange: Evolutionary and Transformative Effects," in Bruce Kogut, ed., *The Global Internet Economy*. Cambridge: MIT Press: PAGES.

Henderson, Rebecca M., and Kim B. Clark. 1990. "Architectural Innovation: the Reconfiguration of Existing Product Technologies and the Failure of Established Firms," *Administrative Science Quarterly* **35**(1): 9-30 (March).

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: 251-74.

Krichel, Thomas, and Christian Zimmermann. 2005. "The Economics of Open Bibliographic Data Provision," Working paper 2005-01, University of

Connecticut, Storrs, Department of Economics. Available at: ‹**http://ideas.repec.org/p/uct/uconnp/2005-01.html**›.

Kuan, Jennifer W. 2001. "Open-source Software as Consumer Integration into Production," Working Paper (January). Available at: ‹**http://ssrn.com/abstract=259648**›.

Kuhn, Thomas S. 1970. *The Structure of Scientific Revolutions.* Chicago: The University of Chicago Press, second edition.

LaMonica, Martin. 2004. "Breaking the Rules with Open-source," CNET News.com (August 2, 4:00 AM PT) ‹**http://zdnet.com.com/2100-1104_2-5290983.html**›

Langlois, Richard N. 1988. "Economic Change and the Boundaries of the Firm," *Journal of Institutional and Theoretical Economics* **144**(4): 635-657 (September).

Langlois, Richard N. 2002. "Modularity in Technology and Organization," *Journal of Economic Behavior and Organization* **49**(1): 19-37 (September).

Langlois, Richard N. 2003. "The Vanishing Hand: The Changing Dynamics of Industrial Capitalism," *Industrial and Corporate Change* **12**(2): 351-385 (April).

Langlois, Richard N., and Paul L. Robertson 1992. "Networks and Innovation in a Modular System: Lessons from the Microcomputer and Stereo Component Industries," *Research Policy* **21**(4): 297-313 (August).

Langlois, Richard N., and Paul L. Robertson. 1995. *Firms, Markets, and Economic Change: A Dynamic Theory of Business Institutions*. London: Routledge.

Lerner, Josh, and Jean Tirole 2002. "Some Simple Economics of Open-source," *Journal of Industrial Economics* **50**(2): 197-234 (June).

McCloskey, D. N. 1995. "Kelly Green Golf Shoes and the Intellectual Range from M to N," *Eastern Economic Journal* **21**(3): 411-414 (Summer).

Miller, Mark S., and K. Eric Drexler. 1988. "Markets and Computation: Agoric Open Systems," in Bernardo Huberman (ed.), *The Ecology of Computation*. Amsterdam: North-Holland: 133-176. Also available at: ‹http://www.agorics.com/agorpapers.html›.

Moore, Gordon. 1965. "Cramming More Components onto Integrated Circuits,"*Electronics* **38**: 114-117 (April 19).

Murdock, Ian. 1994a. “The Debian Distribution,” *Linux Journal* 1es: Article No. 11(March).

Murdock, Ian. 1994b. “Overview of the Debian GNU/Linux System,” *Linux Journal* 6es: Article No. 15(October).

Murdock, Ian 1994c. “The Open Development of Debian,” *Linux Journal* 3es: Article No. 7(June-July).

Murdock, Ian 2003. “Debian: A Brief Retrospective,” ‹http://www.linuxplanet.com/linuxplanet/editorials/4959/1/›.

Nelson, Richard R., and Sidney G. Winter 1977. “In Search of Useful Theory of Innovation,” *Research Policy* **6**(1): 36-76 (January).

North, Douglass C. 1981. *Structure and Change in Economic History*. New York: W. W. Norton.

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): 1053-58 (December).

Parnas, David Lorge, Paul C. Clemens, and David M. Weiss 1985. “The Modular Structure of Complex Systems,” *IEEE Transactions on Software Engineering* **11**(3): 259-266 (March).

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. Also online: ‹**http://www.catb.org/~esr/writings/cathedral-bazaar/**›.

Rosenberg, Nathan 1976. *Perspectives on Technology*. New York: Cambridge University Press.

Savage, Deborah A. 1994. “The Professions in Theory and History: the Case of Pharmacy,” *Business and Economic History* **23**(2): 130-160 (Winter).

Shapiro, Carl, and Hal R. Varian. 1998. *Information Rules: A Strategic Guide to the Network Economy*. Cambridge: Harvard Business School Press.

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).

Simon, Herbert A. 1951. "A Formal Theory of the Employment Relationship," *Econometrica* **19**(3): 293-305 (July).

Simon, Herbet A. 2002. "Near Decomposability and the Speed of Evolution," *Industrial and Corporate Change* **11**(3): 587-99 (June).

Smith, Adam 1976. *An Enquiry into the Nature and Causes of the Wealth of Nations.* Glasgow edition. Oxford: Clarendon Press.

Teece, David 1986. "Profiting from Technological Innovation: Implications for Integration, Collaboration, Licensing, and Public Policy," *Research Policy* **15(**6): 285-305 (December).

Thompson, Clive 2004. "Art Mobs: Can an Online Crowd Create a Poem, a Novel, or a Painting?," *Slate* (July 21). ‹**http://slate.msn.com/id/2104087/**›.

Torvalds, Linus 1999. "The Linux Edge," in DiBona, Chris, Sam Ockman, and Mark Stone (eds.), *Open-sources: Voices from the Open-source Revolution*. Sebastopol, CA: O'Reilly & Associates, Inc.: 101-11. Also online: ‹**http://www.oreilly.com/catalog/opensources/book/toc.html**›.

Ulrich, Karl 1995. "The Role of Product Architecture in the Manufacturing Firm," *Research Policy* **24**(3): 419-440 (May).

von Hippel, Eric 1989. "Cooperation Between Rivals: Informal Know-how Trading," in Bo Carlsson (ed.), *Industrial Dynamics: Technological, Organizational, and Structural Changes in Industries and Firms.* Dordrecht: Kluwer Academic Publishers: PAGES.

von Hippel, Eric, and Georg von Krogh 2003. "Open-source Software and the 'Private-Collective' Innovation Model: Issues for Organization Science," *Organization Science* **14**(2): 209-223 (March-April).

Watts, Duncan 2004. "Decentralized Intelligence: What Toyota Can Teach the 9/11 Commission about Intelligence Gathering," *Slate* (August 5). ‹**http://slate.msn.com/id/2104808**›

Wheeler, David A. 2003. "Why Open-source Software/Free Software (OSS/FS)? Look at the Numbers!," (September 8). Available at: ‹**http://www.dwheeler.com/oss_fs_why.html**›.

Wheeler, David A. n.d.1. “Open-source Software/Free Software (OSS/FS) References.” Available at: ‹**http://www.dwheeler.com/oss_fs_refs.html**›.

Wheeler, David A. n.d.2. “Make your Open-source Software GPL-Compatible. Or Else.” Available at: ‹**http://www.dwheeler.com/essays/gpl-compatible.html**›.