# The Social Context of Open-Source Software

It is truly written: the best hacks start out as personal solutions to the author's everyday problems, and spread because the problem turns out to be typical for a large class of users. This takes us back to the matter of rule 1, restated in a perhaps more useful way:

18. To solve an interesting problem, start by finding a problem that is interesting to you.

So it was with Carl Harris and the ancestral popclient, and so with me and fetchmail. But this has been understood for a long time. The interesting point, the point that the histories of Linux and fetchmail seem to demand we focus on, is the next stage—the evolution of software in the presence of a large and active community of users and co-developers.

In The Mythical Man-Month, Fred Brooks observed that programmer time is not fungible; adding developers to a late software project makes it later. As we've seen previously, he argued that the complexity and communication costs of a project rise with the square of the number of developers, while work done only rises linearly. Brooks's Law has been widely regarded as a truism. But we've examined in this essay an number of ways in which the process of open-source development falsifies the assumptionms behind it— and, empirically, if Brooks's Law were the whole picture Linux would be impossible.

Gerald Weinberg's classic The Psychology of Computer Programming supplied what, in hindsight, we can see as a vital correction to Brooks. In his discussion of ``egoless programming'', Weinberg observed that in shops where developers are not territorial about their code, and encourage other people to look for bugs and potential improvements in it, improvement happens dramatically faster than elsewhere. (Recently, Kent Beck's `extreme programming' technique of deploying coders in pairs looking over one anothers' shoulders might be seen as an attempt to force this effect.)

Weinberg's choice of terminology has perhaps prevented his analysis from gaining the acceptance it deserved—one has to smile at the thought of describing Internet hackers as ``egoless''. But I think his argument looks more compelling today than ever.

The bazaar method, by harnessing the full power of the ``egoless programming'' effect, strongly mitigates the effect of Brooks's Law. The principle behind Brooks's Law is not repealed, but given a large developer population and cheap communications its effects can be swamped by competing nonlinearities that are not otherwise visible. This resembles the relationship between Newtonian and Einsteinian physics—the older system is still valid at low energies, but if you push mass and velocity high enough you get surprises like nuclear explosions or Linux.

The history of Unix should have prepared us for what we're learning from Linux (and what I've verified experimentally on a smaller scale by deliberately copying Linus's methods [[EGCS])](http://www.catb.org/%7Eesr/writings/cathedral-bazaar/cathedral-bazaar/ar01s14.html#EGCS). That is, 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 from hundreds (perhaps thousands) of people.

But the traditional Unix world was prevented from pushing this approach to the ultimate by several factors. One was the legal contraints of various licenses, trade secrets, and commercial interests. Another (in hindsight) was that the Internet wasn't yet good enough.

Before cheap Internet, there were some geographically compact communities where the culture encouraged Weinberg's ``egoless'' programming, and a developer could easily attract a lot of skilled kibitzers and codevelopers. Bell Labs, the MIT AI and LCS labs, UC Berkeley—these became the home of innovations that are legendary and still potent.

Linux was the first project for which a conscious and successful effort to use the entire world as its talent pool was made. I don't think it's a coincidence that the gestation period of Linux coincided with the birth of the World Wide Web, and that Linux left its infancy during the same period in 1993–1994 that saw the takeoff of the ISP industry and the explosion of mainstream interest in the Internet. Linus was the first person who learned how to play by the new rules that pervasive Internet access made possible.

While cheap Internet was a necessary condition for the Linux model to evolve, I think it was not by itself a sufficient condition. Another vital factor was the development of a leadership style and set of cooperative customs that could allow developers to attract co-developers and get maximum leverage out of the medium.

But what is this leadership style and what are these customs? They cannot be based on power relationships—and even if they could be, leadership by coercion would not produce the results we see. Weinberg quotes the autobiography of the 19th-century Russian anarchist Pyotr Alexeyvich Kropotkin's Memoirs of a Revolutionist to good effect on this subject:

Having been brought up in a serf-owner's family, I entered active life, like all young men of my time, with a great deal of confidence in the necessity of commanding, ordering, scolding, punishing and the like. But when, at an early stage, I had to manage serious enterprises and to deal with [free] men, and when each mistake would lead at once to heavy consequences, I began to appreciate the difference between acting on the principle of command and discipline and acting on the principle of common understanding. The former works admirably in a military parade, but it is worth nothing where real life is concerned, and the aim can be achieved only through the severe effort of many converging wills.

The ``severe effort of many converging wills'' is precisely what a project like Linux requires—and the ``principle of command'' is effectively impossible to apply among volunteers in the anarchist's paradise we call the Internet. To operate and compete effectively, hackers who want to lead collaborative projects have to learn how to recruit and energize effective communities of interest in the mode vaguely suggested by Kropotkin's ``principle of understanding''. They must learn to use Linus's La[w.[SP]](http://www.catb.org/%7Eesr/writings/cathedral-bazaar/cathedral-bazaar/ar01s14.html#SP)

Earlier I referred to the ``Delphi effect'' as a possible explanation for Linus's Law. But more powerful analogies to adaptive systems in biology and economics also irresistably suggest themselves. The Linux world behaves in many respects like a free market or an ecology, a collection of selfish agents attempting to maximize utility which in the process produces a self-correcting spontaneous order more elaborate and efficient than any amount of central planning could have achieved. Here, then, is the place to seek the ``principle of understanding''.

The ``utility function'' Linux hackers are maximizing is not classically economic, but is the intangible of their own ego satisfaction and reputation among other hackers. (One may call their motivation ``altruistic'', but this ignores the fact that altruism is itself a form of ego satisfaction for the altruist). Voluntary cultures that work this way are not actually uncommon; one other in which I have long participated is science fiction fandom, which unlike hackerdom has long explicitly recognized ``egoboo'' (ego-boosting, or the enhancement of one's reputation among other fans) as the basic drive behind volunteer activity.

Linus, by successfully positioning himself as the gatekeeper of a project in which the development is mostly done by others, and nurturing interest in the project until it became self-sustaining, has shown an acute grasp of Kropotkin's ``principle of shared understanding''. This quasi-economic view of the Linux world enables us to see how that understanding is applied.

We may view Linus's method as a way to create an efficient market in ``egoboo''—to connect the selfishness of individual hackers as firmly as possible to difficult ends that can only be achieved by sustained cooperation. With the fetchmail project I have shown (albeit on a smaller scale) that his methods can be duplicated with good results. Perhaps I have even done it a bit more consciously and systematically than he.

Many people (especially those who politically distrust free markets) would expect a culture of self-directed egoists to be fragmented, territorial, wasteful, secretive, and hostile. But this expectation is clearly falsified by (to give just one example) the stunning variety, quality, and depth of Linux documentation. It is a hallowed given that programmers hate documenting; how is it, then, that Linux hackers generate so much documentation? Evidently Linux's free market in egoboo works better to produce virtuous, other-directed behavior than the massively-funded documentation shops of commercial software producers.

Both the fetchmail and Linux kernel projects show that by properly rewarding the egos of many other hackers, a strong developer/coordinator can use the Internet to capture the benefits of having lots of codevelopers without having a project collapse into a chaotic mess. So to Brooks's Law I counter-propose the following:

19: Provided the development coordinator has a communications medium at least as good as the Internet, and knows how to lead without coercion, many heads are inevitably better than one.

I think the future of open-source software will increasingly belong to people who know how to play Linus's game, people who leave behind the cathedral and embrace the bazaar. This is not to say that individual vision and brilliance will no longer matter; rather, I think that the cutting edge of open-source software will belong to people who start from individual vision and brilliance, then amplify it through the effective construction of voluntary communities of interest.

Perhaps this is not only the future of open-source software. No closed-source developer can match the pool of talent the Linux community can bring to bear on a problem. Very few could afford even to hire the more than 200 (1999: 600, 2000: 800) people who have contributed to fetchmail!

Perhaps in the end the open-source culture will triumph not because cooperation is morally right or software ``hoarding'' is morally wrong (assuming you believe the latter, which neither Linus nor I do), but simply because the closed-source world cannot win an evolutionary arms race with open-source communities that can put orders of magnitude more skilled time into a problem.