Understanding Open Source Design: A White Paper

In the Beginning Was the Noösphere: Community and Collaboration in Open Source Evolution of Technology

Richard Doyle
Professor of English and STS
Erick Froede
Senior in Mechanical Engineering
David Saint John
Ph D Candidate in Material Science and Engineering
Richard Devon
Professor of Engineering Design
The Pennsylvania State University

Abstract
This paper seeks to codify what we know about open (source) design with a view to using this knowledge to develop and evolve new courses and projects with these methods. We take open source to be a transparent, collaborative process for developing public knowledge that is free at the source, whether for a design or otherwise. The most well known examples are Linux and Wikipedia, but there are many enterprises that qualify such as the Open Source Initiative, the Electronic Frontier Foundation, YouTube, Scribd, Creative Commons and Firefox. Here we will tell the story around the case of Linux.[1]

The paper maps the philosophy and concepts of Open Design,[2] linked historically to a distinct social philosophy frequently dubbed libertarian in the United States but which has equal ties to anarchist practices such as Autonomia in Italy and green anarchism in general. The state in these traditions is often viewed as an obstacle to innovation, posing interesting questions for fields such as nanotechnology that are heavily reliant on grant funding from government agencies. The history of science and technology has long been implicitly open source, with explicit value placed on the free exchange of knowledge and information even while some discoveries, of course, remained closely held secrets. As Robert K Merton famously argued, scientists are bound by "the common ownership of scientific discoveries, according to which scientists give up intellectual property in exchange for recognition and esteem."[3]

Notions of the common good are as frequently found in Open Design as in Utilitarian[4] thinking, but we are also very interested in the degree to which Open Design creates particularly robust systems, has very productive social relations, and is more likely to meet human needs directly than, say, free market or state capitalism (communism) which rely on surrogate mechanisms rather than participatory design.

Introduction

Open source design has grown to prominence over the last few years, gaining legitimacy through products and methodology in many fields, particularly technology. Specifically, software such as Linux, and popular web-based frameworks like Wikipedia, have entered the popular consciousness on a worldwide scale. This growth was explosive, as shown by its presence in academic literature; in fact, "in the three year period 1995-1998 'open source' was mentioned only 12 times…in 2009 the term appeared 687 times, representing an impressive seventeen-fold growth in a decade"[5]

What is the source of this unprecedented expansion? One reason is an estimated 444.8% growth in Internet usage across the globe, with 141.6% in the United States alone.[6] Access, however, does not necessarily justify the increase in participation that has been demonstrated in recent history. According to Levine and Prietula, there are a number of benefits inherent in open source that serve as possible motivators:

> "First is the ability to build upon others' work in the most direct way because the architecture of the good is visible and openly available, such as the lines of code in software or the circuits of a mobile phone. Second, open source may enhance innovation and problem solving by 'removing barriers to entry to non-obvious individuals', allowing users to contribute as much or as little as they may wish (or able) to contribute, without requiring upfront commitment or pre-specified roles…"[7]

Despite the obvious advantages of open source, it is important to ask why this movement actually succeeded. In a world where collaboration is difficult enough in real life, it is surprising to realize how well this movement has taken to the Internet, which has become its natural home. In particular, the internet is "largely devoid of 'the social signaling, cues, and relationships that tend toward moderation in the absence of law'…these were thought to be necessary for such collaborative sharing behavior to occur" (Levine and Prietula). In fact, according to these authors:

> "The same conditions that give rise to open access and consumption with minimal intellectual property rights should have also caused it to quickly collapse under the weight of free riding, as theoreticians have predicted. It is all the more puzzling because participants in open source effort could have easily regulated the sharing of costs and benefits by restricting access to those able and willing to contribute. Sharing does not *necessitate* open source. Nevertheless, open source seems resistant to free riding and highly skewed contributions."[8]

In *The Cathedral and the Bazaar,* Raymond outlines the basic tenets of open design as evident in the Linux experience with no small sense of wonderment:

> "Linux is subversive. Who would have thought even five years ago (1991) that a world-class operating system could coalesce as if by magic out of part-time hacking by several thousand developers scattered all over the planet, connected only by the tenuous strands of the Internet?
>
> ...

> Linux overturned much of what I thought I knew. I had been preaching the Unix gospel of small tools, rapid prototyping and evolutionary programming for years. But I also believed there was a certain critical complexity above which a more centralized, *a priori* approach was required. I believed that the most important software (operating systems and really large tools like the Emacs programming editor) needed to be built like cathedrals, carefully crafted by individual wizards or small bands of mages working in splendid isolation, with no beta to be released before its time.
>
> Linus Torvalds's style of development- release early and often, delegate everything you can, be open to the point of promiscuity- came as a surprise. No quiet, reverent cathedral-building here, [but-] rather, the Linux community seemed to resemble a great babbling bazaar of differing agendas and approaches (aptly symbolized by the Linux archive sites, who'd take submissions from *anyone*) out of which a coherent and stable system could seemingly emerge only by a succession of miracles.
>
> The fact that this bazaar style seemed to work, and work well, came as a distinct shock. As I learned my way around, I worked hard not just at individual projects, but also at trying to understand why the Linux world not only didn't fly apart in confusion but seemed to go from strength to strength at a speed barely imaginable to cathedral-builders."[9]

Far from "flying apart," Linux has indeed been very successful. The historical timeline on Linux.org is peppered with milestones,[10]

- May 2003 The city of *Munich, Germany* announces that it's switching 14,000 PCs from Windows to Linux
- *Oct 2001 Amazon.com* reveals in a SEC filing that switching to Linux has saved them over $20 million.
- March 2000, A *Netcraft* survey reveals that Apache webserver powers 60% of the World Wide Web
- May 1998 *Google* search engine appears using servers running Linux.

In the Beginning Was the Noösphere

Casting evolutionary biology in technological terms, open source development of software (and hardware, as DNA seems to lie somewhere between the two) is a process which seems omnipresent in the development of complex living systems. These biological metaphors are equally applicable to the development of modern software architecture. The roots of Linux extend back to 1969 with a collaborative project between MIT, AT&T, Bell Labs and GE called "Multics."[11] The ethos of knowledge sharing and collaboration through correspondence extend back, of course, much further: Many scientific minds throughout history have worked by exchanging letters and copies of papers between each other, along with the public papers recited

orally to similarly minded scientists, we have some evidence of even older information sharing: thus the recent find of a Babylon instructable for the preparation of beer which is now the oldest recipe in the world. In terms of the overall timeline of knowledge production, open source is likely to be the "old time religion" - the source for each and every development method and mode of inquiry since. Indeed, 3.5 billion years ago, bacteria began engaging in "quorum sensing" behavior - the "promiscuous" swapping of DNA, a primordial code sharing echoed in the later endosymbiotic evolution of the nucleated cell through the "long bacterial embrace" of an invading bacteria and its host cell.

Information sharing among humans shares some similarity with this commons formation activity in biological systems. The Soviet biologist Vladimir Vernadksy dubbed the effect of collective human attention on the biosphere the "noösphere,"[12] and in this paper we wish to trace some events in the development of the open source operating system Linux within the heuristic framework of technological evolution, arguing that the roots of current open source methodologies can be usefully understood according to evolutionary models that include the noösphere, as the informational ecology within which technological evolution transpires. Here, collectives form for knowledge sharing and competition in response to the very human attempts to own and control knowledge production to gain utility and prestige. If 1969 seems ancient in terms of technological evolution - we are talking the era of enormous reels of tape and holorith cards - it is also primordial as a well nigh evolutionary methodology. An awareness of these evolutionary aspects to open source development, we would suggest, can help us accept both the inevitable attempts to own the knowledge it yields and give us clues to the paths for future developments that can be at least as transformative of our planet as the Internet, a majority of whose servers currently run Apache, an open source technology.[13]

Evolutionary models, of course, suggest that not every technology or methodology is going to survive in every ecosystem - evolution operates through principles of selection, and Multics was a "doomed" project from the start. Millions of dollars and months behind schedule, the Honeywell project was cut. Ironically, this act of selection was a liberation as two of the team's brightest minds, Ken Thompson and Dennis Ritchie, became free to pursue the spirit of Multics as they wished. Taking the ideas and concepts developed during the project, Ken Thompson achieved in a four week coding session what four major companies were unable to accomplish : He created an operating system, now known as UNIX.[14] After another act of evolutionary selection (the breakup of the telephone monopoly) ATT began selling UNIX rather than giving it away, spurring the development of BSD by Bill Joy. BSD was promptly mired in legal difficulties. UNIX was different from those that came before it based on the design principle adopted by Thompson: Everything can be considered a file.[15]

This simple design principle made simplifying assumptions, creating a universal name space that would allow modular growth, demonstrating the critical role of design philosophy in the

emergence of open source. This assumption would have far reaching consequences for the open source community: a "UNIX-like operating system could be built piecemeal, and others could help with the task by working independently on some of the various components."[16]

Another act of selection in the evolution of Linux influenced Richard Stallman, who in 1983 was on the eve of a momentous decision. Stallman watched as corporate recruiters raided Harvard's AI Lab: Companies such as LMI and Symbolics lured away the hacker community from the AI lab with (comparatively) lucrative contracts. The AI lab had been both a home and a community to Stallman, fostering an attitude of playfulness and collaboration during his time there that had become ingrained into his character. This was evident even in 1975, when Stallman was working on a text editor through the lab called EMACS, and had established an "informal rule that anyone making improvements had to send them back to him"[17] cultivating a fundamental practice of what Stallmann would later call Free Software. When the AI Lab was left as a shell of its former self due to corporate rivalry, Richard decided to take things into his own hands and keep his ideals alive by creating a new, UNIX-like operating system. Again, an act of evolutionary selection (a corporate raid) yielded innovation, and this innovation was based on two tactics often seen in evolutionary systems: mimicry and stigmergy. That GNU (the recursively named operating system Stallman would develop out of his desire for a printer driver) modeled itself on UNIX is not controversial, despite it's name: GNU's Not Unix. And "stigmergy" names the evolutionary method adopted by termites in the collective building of their mounds: they work with already existing order and apply simple rules. So too did Stallman, in act of evolutionary bricolage,[18] work with the already existing framework.

Stallman treated UNIX as his model for several reasons, but specifically cited its hardware portability and established knowledge base as key factors. Both portability and the widespread knowledge of UNIX, of course, were linked to its simple rules that enabled distributed rather than centralized production: "Everything is a file" means that you have an implicit protocol for relating any new part of the operating system to any other, and "Every file has a single purpose" yielded the modular aspect of the operating system, allowing it to operate as "small pieces loosely joined."[19] However, building this new system now known as GNU was an essentially multiplayer game, and as Stallmann explained "I could write the code, but I couldn't test everything myself. Users had to find the bugs."[20] So perhaps GNU adds a third design principle to UNIX's first two: It takes a village to find and fix the bugs...From this point on, Stallman was committed to distribute GNU as widely as possible, even going so far as developing the GNU EMACS General Public License in 1985, a key legal document in the open source movement and a model for others that followed. It enforced a "copyleft", which enshrined Stallman's values of free distribution and modification, and had the consequence of ensuring the source code for the software would always be freely available. Essentially, this GPL served as a "written constitution...that enabled that world to progress far more efficiently than it had in the past where all these 'laws' were unwritten. As the use of the GNU GPL became more common, so its power

grew, because the more copylefted programs there are the greater the pool for future such programs to draw on; this eases their creation and the pool of GPL'd software grows yet further."[21]

The news indeed spread about GNU. An unassuming Finnish student named Linus Torvalds wanted to begin experimenting with a new operating system of his own. Yet Torvalds could not pursue GNU as his testbed - its kernel was not yet complete. And he quickly discovered that UNIX wasn't the answer, as its availability was extremely limited at his university. Luckily, he discovered an educational operating system called MINIX,[22] created by Andrew Tennenbaum from the Free University of Amsterdam. MINIX was a teaching tool for demonstrating the nuts and bolts of operating systems to students, and was the perfect proving grounds for Torvalds to test some of the ideas he had been thinking about for some time. Minix itself was a response to ATT's decision to restrict access to the source code of UNIX. Andrew Tanenbaum writes:

> I was teaching a course on operating systems and using John Lions' book on UNIX Version 6. When AT&T decided to forbid the teaching of the UNIX internals, I decided to write my own version of UNIX, free of all AT&T code and restrictions, so I could teach from it. . . I set out to write a minimal UNIX clone, MINIX, and did it alone. The code was 100% free of AT&T's intellectual property. The full source code was published in 1987 as the appendix to a book, *Operating Systems: Design and Implementation*,[23] which later went into a second edition co-authored with Al Woodhull. MINIX 2.0 was even POSIX-conformant. Both editions contained hundreds of pages of text describing the code in great detail. A box of 10 floppy disks containing all the binaries and source code was available separately from Prentice Hall for $69.[24]

While the software was not yet free, MINIX was a bargain compared to a 300 hundred dollar educational UNIX license. More importantly, the "internals" were available for teaching and exploration, and it was this opening of the source code that would enable Torvald's new project: a UNIX like operating system for the new Intel 386.

By September 1991, Torvalds had assembled what was to become the first version of his operating system, dubbed version 0.01. Even at this early stage there was interest in the community to help with this grand new venture, as after Linus had declared his intentions on the 40,000 member MINIX discussion board another user from Austria responded "I have already thought of writing my own OS, but I decided I wouldn't have time to write everything from scratch. But I guess I could find the time to help raise a baby OS!" This "baby OS" demonstrated the strength of the open source model to collectively reproduce itself, fulfilling Stewart Brand's axiom that "Information wants to be free."[25] It was a small scale release, only involving about ten individuals who responded to Torvalds posting to the os.minix USENET Group. The initial release "forbade anyone from profiting financially" from it,[26] but by version

.012, Torvalds invoked the GNU copyleft originally defined by Stallman. By 1994 users and contributors had grown exponentially.

By involving the community from the start and staying true to open source foundations (namely the GNU GPL), Torvalds had taken the first steps to forming open source as we know it today. The highly competitive realm of programming became the collective enterprise of Open Source development, and the shared code base of Linux became the product of thousands of developers working together at a distance. Kernel 2.6.2, for example, had "thousands of developers - at least, almost 2,000 who put in at least one patch."[27]

This collective generation of value represents well the remarkable capacities of hobbyists - Torvald's term for himself in his originally declaration of Linux post as well as Bill Gates' original status in the Home Brew Computer club - in generating leading edge information technology. RepRap,[28] an open source 3-D printer project with the goal of self-replication of basic components, reminds us that this productivity of hobbyists remains a potent force in technological evolution. The current small trickle of listserve and wikispace dedicated to this 'garage or basement capable' 3-D printing technology may very well serve to create similarly crowd-sourced developments in the field of localized manufacture and perhaps sustainability more generally. This is important to remember in the context of engineering culture and the marketing of technology that emphasizes the expertise of specialists and experts. Torvalds would later have a highly public debate with Tanenbaum,[29] the author of Minix, in which Tanenbaum explicitly treats Torvalds as an "F" student even as Torvalds was creating a new operating system. Part of what challenges conventional paradigms and business models in open source is the notion of bottom-up expertise. With collective intelligence enabled by the internet, it is no longer only the experts who can make a contribution to technological evolution, and the "F" students can outflank the professors, given some inspired insight into a new mode of operation. This flies in the face of institutional expectations of expertise through authority and credentialing, but sometimes proves to be a viable life strategy nonetheless.

Perhaps even more radically, Linux and other successful open source projects suggest the plausibility of gift culture out competing global capitalism and its mechanisms of ownership and the "digital enclosure" of Digital Rights Management. Eric Raymond, an open source advocate whose text "The Cathedral and the Bazaar"[30] was seminal in imagining the highly distributed and less centralized forms of production augured by open source, noted that this culture of sharing and gifting had much in common with academic practices, suggesting a "pattern" in the production of creative work:

> One might suspect that academia and the hacker culture share adaptive patterns not because they're genetically related, but because they've both evolved the one most optimal social organization for what they're trying to do, given the laws of nature and the

> instinctive wiring of human beings. The verdict of history seems to be that free-market capitalism is the globally optimal way to cooperate for economic efficiency; perhaps, in a similar way, the reputation-game gift culture is the globally optimal way to cooperate for generating (and checking!) high-quality creative work.[31]

In other words, there is nothing natural about the competitive advantages offered by global capitalism in all domains of work. In some, such as open source development, it is gift culture that is the "globally optimal way" to model and organize the creative collaborative work of decentralized "clouds" of human beings. How are we to understand this advantage of a gift culture in a world that allegedly values labor only in terms of its worth on a market?

Raymond suggests that there is an implicit market or "reputation-game" that has more in common with Darwinian sexual selection ("survival of the sexiest") than it does with the survival of the fittest paradigm of corporate competition. Here hackers compete for "brand" value associated with their signature in a competition for prestige. In the case of Torvalds, it would seem to have been as much of a "curiosity-game" as a reputation-game, but under each of these models we can glimpse a system of development linked not to economic success but to a "world of giving". The incredible efficacy of gift culture in generating value - Linux has itself been valued by economists at well over 10 billion (Linux Foundation, 2008),[32] and the value of Wikipedia can only be compared to the valuation of its closest competitor, Encyclopedia Britannica - suggests that the next wave of technological development, nanotechnology, could similarly profit from practices of gift culture.

What Nano Can Learn From Linux

Nanotechnology, at first blush, would seem to be an unlikely candidate for open source development: It is capital intensive, reliant upon expensive instrumentation such as electron microscopes and associated characterization technologies, and the input of relevant experts and Nobel Laureates. Yet costs of instrumentation are plunging, with desktop Atomic Force Microscope (AFMS) fetching around 25,000 USD and networks such as NNIN offering diverse services at a distance for lower and lower costs. Some nano-facilitating techniques such as microfluidics and emerging desktop manufacturing produce highly shareable information and replicatable hardware. A hobbyist culture associated with DIY biology has already emerged, and websites and kits featuring DIY nano are sprouting up associated with initiatives such as Open Science.[33] Combine this ability for highly distributed, "bazaar" style production and exchange with the promises of nanofiltration[34] (water for a billion people), nanoenabled photovoltaics[35] (electricity for two billion people), or arsenic removal[36] and you have the outlines of a gift culture where participants compete for the prestige of most effective techno-altruist. As contemporary science research (nano or not) in the US is overwhelmingly financed by government grants, the question of "gifting" and sharing of emerging nanotechnological knowledge towards the bootstrapping of sustainable culture approaching a baseline global equity is hardly a pipe dream -

in some sense it is the main means through which our new scientific knowledge is currently generated already. A grant is a gift with expectations attached. Should the knowledge produced with such grants be subject to the hoarding of intellectual property, e.g. patents?

So what are the features of such a gift culture as suggested by the evolution of Linux?

1. Bottom up innovation and production: Innovation may not come only from "A" students, but also from "F" students like Torvalds. This suggests at the very least that undergraduate students can be an important resource in development of fresh ideas.
2. Plenty of intercultural conflict: Torvalds and Tennenbaum notoriously engaged in a brief but intense "flame war", and current open source projects such as OpenBSD continue to feature this collective unity that emerges from conflict.
3. Corporate Buy-in: Good data are hard to come by, but most of it suggests that Linux enjoys a goodly amount of corporate support in the form of direct and indirect sponsorship of employees who do significant work in Linux development. IBM and Google have both made major contributions. There is no necessary antagonism between gift culture and corporate culture.
4. Implicit Incentives: The drivers for this creative technological evolution are not the obvious ones of money and power. Prestige and reputation can generate billions of dollars of value (Linux, Wikipedia, Gates Foundation). Branding, and the marketing thereof, thrive with this type of image as it withstands scrutiny by being genuine.
5. Explicit Goals: Reputation-games only seem to work with enormous transparency and well defined targets.
6. Global Standards Enable Open Source: One of Torvald's first moves in developing Linux was to request access to the POSIX standards extant for UNIX. In nano, global standards can create protocols that are a similar substrate for further development.

These heuristic principles suggest that it is the classroom itself which could serve as an incubator for open source nanotech research. Penn State's emerging curriculum in nanotechnology and its social, ethical and legal impacts has worked best with a mixture of graduate and undergraduate students and a flat pedagogical hierarchy. *We need to select against the culture of top down expertise in this multidisciplinary and rapidly developing field.* Outside of the classroom, a nascent student club (the Intercollegiate Futures Society) cultivates the hobbyist ethos with hands on work with a Rep-Rap Mendel, (the aforementioned open source 3-D printer). It also increases the range of majors that get recruited into nanotech related work in policy, scenario planning, ethics, and commercialization. We're hoping we can find a few good "F" students like Torvalds to help us out.

Bibliography

1. Linux online http://www.linux.org/ Viewed 10/4/2010
2. Robert K Merton Wikipedia Viewed 10/4/2010
3. Open Design Wikipedia Viewed 10/4/2010
4. Utilitarianism Wikipedia Viewed 10/4/2010
5. Levine, Sheen S. and Prietula,Michael J. "Where and When Can Open Source Survive? Towards a Theory of Robust Performance "*IFIP Advances in Information and Communication Technology*, 2010, Volume 319/2010, 156-176.
6. Internet World Stats http://www.internetworldstats.com/stats14.htm Viewed 10/4/2010
7. Op. cit.
8. Op. cit.
9. Raymond, Eric S. *The Cathedral and the Bazaar.* First written in 1997 and the full book is available here http://catb.org/esr/writings/cathedral-bazaar/
10. Linux Online. A History http://www.linux.org/info/linux_timeline.html Viewed 10/4/2010
11. History of Multics. http://www.multicians.org/history.html Viewed 10/4/2010
12. Noosphere. Wikipedia Viewed 10/4/2010
13. Apache HTTP Server Project. http://httpd.apache.org/ Viewed 10/4/2010
14. Ken Thompson Wikipedia http://en.wikipedia.org/wiki/Ken_Thompson Viewed 10/4/2010
15. "In Unix Everything is a File" http://ph7spot.com/musings/in-unix-everything-is-a-file
16. Moody, Glyn. Rebel Code: Linux and the Open Source Revolution Basic Books, 2002.,p14.
17. Op cit p17
18. Jacob F.1977. Molecular tinkering in evolution. In: Bendall DS, ed. *Evolution from molecules to men.* Cambridge: Cambridge University Press.
19. O'Reilly, Tim. "The War for the Web" ttp://radar.oreilly.com/2009/11/the-war-for-the-web.html Accessed October 5, 2010
20. Moody, op cit., p 19
21. Moody, op cit., p p 27-8
22. Minix. http://www.minix3.org/
23. Tanenbaum, Andrew S. and Woodhill, Albert S. Operating Systems: Design and Implementation 1997.
24. Tannenbaum, Andrew. http://www.cs.vu.nl/~ast/brown/ Viewed 10/4/2010
25. Brand, Stewart. "The Media Lab: Inventing the Future at MIT". Penguin Books, 1989, p201
26. The Internet: An Historical Encylopedia, p. 233
27. Corbet. "Who wrote 2.6.20?" 2007. http://lwn.net/Articles/222773/ Accessed October 5, 2010
28. RepRap http://reprap.org/wiki/Main_Page V iewed 10/4/2010
29. Tanenbaum-Torvalds debate http://oreilly.com/catalog/opensources/book/appa.html Viewed 10/4/2010
30. Raymond, op cit.
31. http://catb.org/esr/writings/homesteading/homesteading/ar01s19.html Viewed 10/4/2010
32. Linux Foundation. "Linux Foundation Estimates the Value of Developing a Distribution of Linux" Accessed October 5, 2010.
33. Open Science. http://www.openscience.org Accessed October 5, 2010.
34. Nanofiltration. SDWF http://www.safewater.org/PDFS/resourcesknowthefacts/Ultrafiltration_Nano_ReverseOsm.pdf
35. Nano Frontiers: Nanotechnology: Energizing the Future. NanotechProject.org 2008
36. M. Lounsbury et al., "Towards Open Source Nano: Arsenic Removal and Alternative Models of Technology Transfer," *JAI Advances in the Study of Entrepreneurship, Innovation, and Economic Growth* 19 (2009): 51-78. DOI:10.1108/S1048-4736(2009)0000019003