The Internet in the Open Source Age

Randall S. Wood

20th May 2004

1 Introduction

> "How the code regulates, who the code writers are, and who controls the code writers – these are questions that any practice of justice must focus on in the age of cyberspace. The answers reveal how cyberspace is regulated . . . If we admire the Net, should not a burden of proof fall on those who would change the basic assumptions [of liberty and openness] that brought it about in the first place? (Lessig 1999)"

Regulation takes many forms: while heavy-handed, top-down government legislation regulation is probably the form that first comes to mind, it is not the only type of regulation that controls the activities of human beings. Rather, rules, norms, technologies, and society itself all contribute to the forces that direct and shape the way humans live and behave. Laws can take the form of copyright protection, defamation law, and the like, while norms govern what behavior is appropriate at a given time (like, for example, smoking in someone else's house, which societal norms indicate is appropriate only if you've asked permission first). Market forces, such as the price of alcoholic beverages, can impose constraints that are interdependent and which can supplant or reinforce the other forces. And lastly, architectural constraints – the design of things itself – are equally as important and relevant as all the other elements that contribute to regulation, including government regulation itself (Lessig 1999).

The Internet is not free from these restrictions of norms, market forces, and so on. In fact, the social and political nature of the Internet is determined by these forces to a greater extent than it is by any one government's regulation. The architecture and software of computers and networks – that is, the code – are perhaps more important than regulatory regimes in determining the social and political nature of the Internet. Whether a web site requires you to input a password before being admitted access, or to interact with other users and the site itself using a psedonym is a conscious decision the author of that web site's software has made; if certain interactions through e-mail require encryption as a mechanism to ensure identity or to obscure the

message's contents, it has been specifically planned that way by the two participants in the engagement. Likewise, if users enable and permit the use of unrequested 'cookies' by web sites and if the web browser facilitates that interaction, the architecture – the form – of the interaction between the user of that web site and its designer have been determined by a conscious agreement in which the software and hardware are simply the mechanism. This is the theme of Lawrence Lessig's *Code and Other Laws of Cyberspace* (Lessig 1999).

Because computer code determines the architecture of the Internet, it can profoundly affect competition and market innovation, shape the role that universities, non-profit organizations and research institutions play, and determine the amount of control users of the Internet have. Who writes the code that governs the Internet, as well as who controls those coders, is thus an essential factor in the Internet's architecture. This makes the Open Source softwarea movement more important than ever (Canali di Rossi 2002).

2 The Open Source Movement

The open source development model may fundamentally change the forces that shape the Internet. In fact, it's tempting to believe that at the beginning of the 21st century the world of software development has come almost full-circle in terms of programming philosophy. Open source software (alternatively called Free software or FOSS – free and open source software) has existed throughout the history of software development. In fact, in the early days of programming, all software was essentially open source with no real need to give it a name: programmers shared their code with each other to make advances and challenge the machines on which they worked. It was science and intellectual challenge, not business, and no one would ever think of denying another programmer the right to see how his program functioned. Over time, that philosophy changed.

Richard Stallman, the philosophical leader of the open source movement, was inspired to push for a return to sharing and cooperation in a highly circumspect way: by struggling with an early Xerox laser printer. He

and co-workers had modified an earlier printer connected to their PDP-11 machine by changing the software to check periodically on its availability and report back to the server, and in the case of printer jams, to send a message to users requesting the jam be fixed so printing could continue. It wasn't a part of the original software for the printer but because they had access to the source code for the printer's interface they were able to implement the change. But the new Xerox machine was built under new circumstances, and was provided without source code. When it jammed – which it did, frequently – Stallman and his team realized they were unable to tinker with the machine's instructions. Furthermore, Xerox was unwilling to distribute the code, fearing it would lose its trade secrets. In a sense, software had become valuable in its own right and not simply because it permitted computing machinery to operate, and companies were no longer interested in making their proprietary software public. Reflecting on the slippery slope of licensing, non-disclosure agreements, and proprietary code, Stallman was compelled to dedicate his life to another developing and encouraging another way of doing business, in which the brotherly attitude that had governed computer science and programming since its inception would overrule corporate concerns for protection. The GNU project was born (Williams 2002).

Stallman's own description of the movement and his passion for "freedom" is evocative:

> Free Software is about giving software users the freedoms that are necessary to treat each other as friends and form a community. This means that you must have the freedom to change a piece of software to do what you want or need it to do. You must also be free to redistribute that software so you can help your neighbor. It follows from there that you must be free to publish an improved version of that software, so that you can share your improvements with other people who can also benefit from it and build on it further. These freedoms provide practical and social benefits, both of which are important (Linux Magazine 1997).

These freedoms impact the Internet in profound ways, by changing the

power relationship between users of software and the institutions with which they interact.

3 Proprietary vs. Open Source approaches to coding

There are many known benefits to the open source model that have been proven with experience: over time and if fully-developed, the open source process leads to innovative, cost-efficient software that is more adaptable, better in quality, more stable (less prone to crashes), and more secure, all at a typically lower cost (Canali di Rossi 2002). In fact, many of the most popular and widely used programs on the Internet are open source software, including BIND (the Berkeley Internet Name Domain, responsible for packet routing), the Apache web server (responsible for nearly 70% web pages served daily) (Netcraft), the Mozilla web browser, and the SAMBA protocol suite that allows non-Windows software to communicate with Windows machines more efficiently than Windows machines themselves are able to do so (Howorth 2003).

But more important than the fact that the open source development model is able to create quality software is its flexibility. Open source software is able to distinctly address the specific needs or atypical situations. Open source software projects represent a huge repository of code which can be utilized as building blocks for software projects. The Rhode Island Department of State, for example, designed a particular database to catalog its immense collection of documents, rules, meeting minutes, and more. A colleague from Hawaii was able to take and adapt the same project for its own needs, saving time and money. Corporations that develop software and contribute it to the open source community find that other users, in contributing to the project, enjoy better quality software that's been better tested and more thoroughly probed (Zetlin 2003). A 2003 study confirmed this systematically, validating what the open source community had been claiming all along: that "many eyes make short work of bugs." In the study, Reasoning,

a company that sells automated software inspection services, scrutinized the TCP/IP stacks of Linux and five other operating systems and found the code exhibited a higher quality than commercial implementations of TCP/IP stacks or two special-purpose networking products (Shankland 2003).

The free nature of open source code provides additional benefits in addition to its quality. To IT managers, open source code means cost savings, as they are given the freedom to patch and upgrade their systems as they see fit. The freedom to upgrade when one sees fit also translates into economic savings. Open source software also provides business opportunities for other companies. When Red Hat Linux decided to stop supporting the older versions of its software a company by the name of Progeny was able to fill Red Hat's shoes, offering annual software updates for $5 per user per year. Because only Microsoft controls the source code for the Windows operating system only Microsoft can provide – not refuse to provide – updates for its older software. Choosing not to do so forces companies into upgrading to newer versions of the Windows operating system, often at great expense: a business with 300 desktops running Red Hat would have paid $1500 to upgrade; a business with the same number of desktops using Windows 98 would have faced $60,000, 40 times higher (Feldman 2004).

The open source software development model is not without its shortcomings, of course, which is simply to point out that every tool has its advantages and disadvantages and it's wise to choose the appropriate tool for the job. Nikolai Bezroukov points out several of the ways in which the open source development model is inappropriate and its aficionados shortsighted in treating it like a panacea (Bezroukov 1999).

First of all, computer programs are complex, and the more complicated they are, the more complex they are. Is there any difference in the opacity of a 10 line binary program (i.e. compiled into 1s and 0s) and a ten million line program with inadequate documentation? In essence, there isn't. While open source software development relies upon large groups of participants involved to different degrees and in different ways, to function, per the "many eyeballs" approach. But in many cases, large organizations make for slow decision-making processes, generate disagreements, and make

organizations slower and more cumbersome. But trying to overcome this through the establishment of an hierarchy or a project elite can generate ill-will and reduce participation. In fact, a high degree of centralization, such as the establishment of a clear and concrete goal, is critical to open source projects. Conflicts over software design, methodology, and other disagreements can lead to stagnation and setbacks, or worse, project forks (in which two projects are spawned from one and each project develops in parallel but differently) (Bezroukov 1999).

In a similar way, the advantage of open source – the fact that so many participants can be involved in so many ways – leads to more experimentation and more duplicated effort. Linux boasts several graphical desktops, for example, among them Gnome, KDE, XFDE, Blackbox, Windowmaker, and others. While they differ from one another in notable ways, they have fragmented the developers into competing camps instead of concentrating on a single effort (on the other hand, this kind of competition has spurred innovation and experimentation in all of them). Lastly, the open development model can lead to the abandonment of some software projects, due to their inability to acquire a critical mass of users, loss of the leading developer, or burn-out. Open source advocates would call this Darwinism, weeding out weaker projects, and would point out that thanks to licenses like the GPL, no project is ever really abandoned, because the code remains available to others who sometime later may grow interested in taking up the project again, or utilizing the code in different projects of their own (Bezroukov 1999).

Jamie Zawinski, one of the original authors of the Netscape Navigator browser – which was later reborn in the form of Mozilla, one of the first and most publicized open source software projects – noted, “Open source does work, but it is most definitely not a panacea. If there’s a cautionary tale here, it is that you can’t take a dying project, sprinkle it with the magic pixie dust of “open source,” and have everything magically work out. Software is hard. The issues aren’t that simple (Zawinski 1999).”

4 Importance of Open Standards

One of the most important effects the open source community has had on computing has been highlighting the importance of open protocols and standards that allow uninhibited communication between software programs at every end of the Internet. Because the Internet was designed in a protocol-focused, application-ambivalent architecture that emphasizes coordinated communication without regard to which kind of appliance or software package is doing the communicating, the implementation and preservation of protocols is crucial to the Internet's architecture. Moreover, protocols are still a point in the Internet ecosystem where regulation could effectively gain traction. Whether a protocol was open or proprietary in this case would be irrelevant (Committee to Study Global Networks and Local Values 2001).

Microsoft itself realized the threat the open source development model posed the company. The memos in which Microsoft documented this perceived threat were leaked to the world in October 1998 in what are now called the "Halloween Documents." In them, Microsoft avers, "OSS poses a direct, short-term revenue and platform threat to Microsoft, particularly in the server space," "Additionally, the intrinsic parallelism and free idea exchange in OSS has benefits that are not replicable with our current licensing model and therefore present a long term developer mind share threat," and "Linux and the other OSS advocates are making a progressively more credible argument that SOS software is at least as robust – if not more – than commercial alternatives. The Internet provides an ideal, high-visibility showcase for the OSS world (Microsoft 1998)."

At the same time, Microsoft hinted at the manner by which it intended to defend itself: "OSS projects have been able to gain a foothold in many server applications because of the wide utility of highly commoditized, simple protocols. By extending these protocols and developing new protocols, we can deny OSS projects entry into the market" and "Linux can win as long as services/protocols are commodities (Microsoft 1998)."

This assertion makes clear why it is neither appropriate nor in the

interest of society to leave the development of the Internet to private corporations, as their fundamental goal conflicts directly with the goals and needs of society at large. Furthermore, it emphasizes Lessig's point: the Internet can be regulated as long as its code is developed in a way that permits it to be controlled. Proprietary protocols put the power in the hands of one company, which a government can then regulate. Open protocols prevent this conundrum.

The battle for protocols was made available in a very real way during the IM battles which began in around 2000. Market leaders AOL and Microsoft fought tooth and nail for customers for their IM (instant messaging) services, changing their own protocols whenever possible to prohibit the other's software from accessing their own client base, while simultaneously trying to prevent the other from doing the same. Those companies that use private protocols to facilitate communication with their software are typically disinclined to share that protocol with competition as it levels the playing field (Festa 1999). But the battles incur bad press, as well. When Netscape Inc. was criticized for manipulating HTML tags to gain advantage over its competition it was forced to respond to the market with a pledge to respect open standards (Homer 1997). The battle between corporations to lock in customers through proprietary extensions to protocols continues, as the anybrowser campaign roundly denounces (Burnstein 2004).

There's reason to believe the value of open source programming has infiltrated the ranks even at Microsoft, where the idea of sharing source code – the company's crown jewels, essentially – has been anathema for years. In 2003 the software giant began a program called "Shared Source" in which certain trusted clients – large enterprises with a certain number of licenses, essentially – were able to receive limited sections of Microsoft source code for their inspection but not modification. This was a major victory for the open source community, as Microsoft had regularly attacked the open source community's GPL license as "viral" (Sims et al. 2001).

Microsoft's shared source initiative is an effort to include the value gained by allowing programmers to examine others' source code and improve upon it, while still enabling businesses to make a profit from their work. This

balance is not easy to strike. Microsoft has chosen to focus on the academic community first, which makes sense, as much academic experimentation with source code happens in that context, but it's a strategic move as well, as Microsoft faces a fierce challenge in the academic world from Java, which enjoys a superior mind share in computer science departments in most universities. But allowing access to some code, while charging for others, is a page taken from the open source community's own book, and could well prove to be an effective method of doing business while simultaneously fostering the communities that foster development, inspiration, and progress (Sims et al. 2001).

5 What Open Source Code Means for Regulation of the Internet

Open source code presents a fundamental problem for those who would regulate the Internet. When code is proprietary its implementation is restricted to the owner of that code. This presents a logical bottleneck where legal restraint can be imposed; for example, if the government mandates telephone companies to implement a certain type of network software, the companies will be forced to comply and the customer will be forced to accept the change, as he/she has no power to fiddle with the software that provides connectivity to the phone system. In this case the fact that the software code is controlled exclusively by the telephone company makes the company itself vulnerable to regulation by the government. But open code, because it belongs to no one in particular, presents no such bottleneck, and as such the forced regulation of a certain implementation of that software is impossible. Software users will always have the freedom to modify that code and will use the freedom to strip out the implementations they don't agree with (Lessig 1999).

The implementation of SSL (secure socket layer) software in web browsers makes a good case in point. When web browsers began to introduce SSL technology in their browsers the French government opposed the

development, hoping to retain the ability to inspect the web transactions as they happened. Because the web browser was already available in source code form, however, it was impossible to make any such mandate, as users had the ability to download the source code and either include or exclude the parts of the software they wanted. Because the code was essentially in the open domain, the ability of the French government to restrict its use was diminished This is a powerful opportunity for those who believe the Net should be free and unregulated: because the target of such regulation is mutable, the ability of a government to impose regulation is diminished. This does not eliminate a government's ability to regulate, but rather changes the nature in which it will do so (Lessig 1999).

Governments are, of course, not without alternatives. One direction corporations will have to turn in order to regulate and even attempt to eliminate open code is by prosecuting its presumed author under national, private legislation such as copyright law and similar legislation. Under pressure from the Motion Picture Association of the U.S.A, the Norwegian government attempted but ultimately failed to convict Jon Johansen for his alleged authorship of software that circumvented the anti-copying mechanisms of commercial DVDs. While Johansen was found innocent on those charges (Online journal theregister.co.uk described the court proceedings as follows: "The entertainment lobby has failed to persuade a Norwegian court to convict a teenager for creating a utility for playing back DVDs on his own computer." (Leyden 2003)), similar lawsuits have been filed against him for his involvement in other software projects that attempt to circumvent encryption technology in digital media (Wikipedia Online Encyclopedia 2004).

Adobe Systems Incorporated, the company that produces Adobe Acrobat and other popular software packages, attacked Dimitry Sklyrov in a similar way for describing the weak encryption Adobe had incorporated into its e-books reader and discussing, for academic reasons, how easily it could be circumvented (Electronic Frontier Foundation). The move to attack supposed authors instead of their companies is one possible, but necessarily inefficient, mechanism by which code can be controlled. But even the lawsuit against

Johansen didn't prevent the code he allegedly authored from being distributed over the Internet: it's available even today.

The existence of open source software then is important to the Internet ecosystem in many ways, but open source software by its open nature alone is not enough to ensure the Internet will remain unregulated; rather open source software will force regulation to take other routes, such as corporate lawsuits against alleged offenders of laws that protect intellectual property, media, or encryption technology; the regulation of the protocols that permit transactions on the Internet, or the adoption of new protocols that facilitate e-commerce. In the latter case, the open source software community may even find itself at a disadvantage because it is dependent on open-, not proprietary- protocols. Open source software is important in other ways however, the most important of which is its ability to lend some diversity to the Internet ecosystem, and to challenge software companies to be more innovative. The open source community benefits from synergies of inspiration and innovation other development models can not provide and there is good reason to believe this enthusiasm will make software code – and thus the Internet – better and more robust. Will the open source movement fundamentally change the Internet? Not likely. But it will certainly protect some of the freedoms the Internet's original founders cherished. And that may be enough.

Notes

[1]Certain Usenet groups are still quite active. While not free of spam and other nuisance posts, they are kept relatively clean by the use of very strong group social pressures to behave properly, bottom- not top-post, and be a good netizen. Posters who are unable to follow the code of etiquette are promptly filtered and ignored. comp.text.tex is the most remarkable example of this social regulation I know of.

[2]This ability remains available to us today. All browsers allow one to view the code and save it as a local file.

References

Bezroukov, N. (1999). Open source software development as a special type of academic research (critique of vulgar raymondism). *First Monday*. <http://www.firstmonday.dk/issues/issue4_10/bezroukov/>.

Burnstein, C. (2004). The any browser campaign. <http://www.anybrowser.org/campaign> Accessed 20 May 2004.

Canali di Rossi, L. (2002, June 30). Open source advantages and limitations. Internet published. <http://www.masternewmedia.org/2002/06/30/open_source_advantages_and_li%mitations_overview.htm>.

Committee to Study Global Networks and Local Values (Ed.) (2001). *Global Networks and Local Values*, pp. 24–35. Washington DC: Computer Science and Telecommunications Board National Research Council.

Electronic Frontier Foundation. Us v. elcomsoft & sklyarov faq. Internet published. <http://www.eff.org/IP/DMCA/US_v_Elcomsoft/us_v_sklyarov_faq.html>.

Feldman (2004, May). The morality of open source software. *Internet Week*. <http://www.internetwk.com/allStories/showArticle.jhtml?articleID=18400%311> Accessed 17 May 2004.

Festa, P. (1999, August 6). Why open standards are a myth. *CNet news.com*. Accessed 17 May 2004.

Homer, M. (1997, June). Netscape open standards guarantee. Accessed 17 May 2004.

Howorth, R. (2003, October 13). Samba beats windows. *VUNet.com*.

Lessig, L. (1999). *Code and Other Laws of Cyberspace*. New York NY: Basic Books.

Leyden, J. (2003, January 7). Dvd jon is free ? official. *The Register*.

Linux Magazine (1997, July). Stallman interview. *Linux Magazine*. <http://www.linux-mag.com/1999-07/stallman_01.html>.

Microsoft (1998). Open source software: A (new?) development methodology (a.k.a. halloween document i). Memo leaked to the Internet. <http://www.opensource.org/halloween/> Accessed 16 May 2004.

Netcraft. April 2004 webserver survey. Internet published. <http://news.netcraft.com/archives/web_server_survey.html> Accessed 20 May 2004.

Shankland (2003, February 19). Study lauds open-source code quality. *Cnet News.com*. <http://msnbc-cnet.com.com/2100-1001-985221.html> Accessed 16 May 2004.

Sims, D. et al. (2001). Microsoft plans shared source .net. *O'Reilly OnDotnet.com*. <http://www.ondotnet.com/pub/a/dotnet/2001/06/27/dotnet.html> Accessed 16 May 2004.

Wikipedia Online Encyclopedia (2004, May). Jon johansen. Wikipedia. <http://en.wikipedia.org/wiki/Jon_Johansen> Accessed 17 May 2004.

Williams, S. (2002). *Free as in Freedom: Richard Stallman's Crusade for Free Software*, Chapter For Want of a Printer. O'Reilly Marcy. <http://www.oreilly.com/openbook/freedom/ch01.html> Accessed 16 May 2004.

Zawinski, J. (1999, March 31). Nomo zilla: resignation and postmortem. <http://www.jwz.org/gruntle/nomo.html>.

Zetlin, M. (2003, April 7). In the linux loop: companies that regularly share code fixes and revisions benefit from more stable and reliable open-source software. *Computerworld*, 37. Lexis Nexus. Accessed 15 May 2004.