

Gabriella Coleman

Opening Remarks given on October 3rd, 2007 at the Brecht Forum for Decoding Liberation by Samir Chopra and Scott Dexter
Routledge, 2007

Providing a brief overview of a book as I am tasked with doing here is a fraught task, marked by paradoxical difficulties. One goes through great pains in writing a book to add flesh onto the skeleton of basic ideas; yet in providing an overview, one must remove the flesh and present the very skelitized version that the book is supposed to leave behind in the closet. Despite this, I hope to provide a synopsis of a book that will motivate those here to read the it, for Decoding Liberation provides an important set of contributions that clarify the stakes involved in the politics of information access as well as the politics of cooptation.

Because there may members of the audience who know very little or nothing about free software, I will first provide a brief overview of the history of free software. Then I will move into what I think are two of Decoding Liberation's most important contributions by briefly visiting the work of two authors: Susan Sontag and Friedrich Nietzsche. Finally, I will leave the author with some questions or comments that I hope will spur further discussion.

What academics, programmers, policy makers, and activists now designate as free and open source software (F/OSS), once had a far simpler name and philosophy, free software—a name and concept born in the halls of MIT under the tutelage and direction of a hacker Richard M. Stallman upset by the incursion of copyrights and patents into software. His official quest to thwart the spread of IP in software began in 1984 and entailed the creation of a foundation (the Free Software Foundation). The most ingenious part of his political response was to use fire – i.e. the law – to fight fire. Stallman created a license, the General Public License, where he

retained copyright in his code, but distributed it freely, provided all of its users did so as well. Free software started to break away from this singular association with Richard Stallman when Linux appeared; Linux is a software project initiated in 1991 by a young Finnish student, Linus Torvalds and he released the source code of his hobby project on a mailing list, using the license RMS created. His project was soon combined with Stallman's GNU software to create what is today widely known as the GNU/Linux operating system. Via his famous volunteer-run project, Linus Torvalds (and somewhat inadvertently) boosted the public image of free software, while Eric Raymond, a libertarian-leaning hacker, sought to explicitly attract more attention, especially those of Silicon Valley venture capitalists. And he did so by excising the language of freedom and rights by replacing the term free software with an ostensibly more neutral terminology of “open source software.”

It is within this basic story of the initial rise of free software and the subsequent appearance of its more “Wall Street friendly” cousin, open source, where *Decoding Liberation* enters and it does so through two distinct but complementary registers: One is by describing the world of free software in all its ethical, aesthetic, and political richness while the second register enters a more critical territory. The authors wear two distinct hats: one as legible computer scientists, making sensible such arcane concepts such as code, computers, and programming, and the other is that of the critical philosopher.

To consider the importance of the first register, I would like to now turn to Susan Sontag for a moment. In a short but biting essay on the state of art criticism, “Against Interpretation” Sontag argues that an excess of critical interpretation, especially the variant that holds “a contempt for appearances” robs us from appreciating what art has to offer. She urges taking a

sharp turn in art criticism so that it can instead “supply a really accurate, sharp, loving description of a work of art” so that we can experience “the luminousness of thing in itself, of things being what they are.”

One of the stunning parts of *Decoding Liberation* is how it provides this “accurate, sharp and even loving description” of free software in multiple capacities, although they place most of their very careful attention on the artifact that binds much of the movement together: software. As they insist in their introduction: “To understand [software] as mere machine instructions, to ignore its creative potential, and its power to enforce political and social control, is to indulge in a problematic blindness.” Because of the importance of software, they open their book with a history of computing and software development, programming languages, the hacker community, and finally examine the changes in political economy that converted software from an open technology into bound and closed commodities, secured as such by the application of copyrights, trade secrets and patents on to software.

But there is one chapter in particular—Free Software and the Aesthetics of Code—that illuminates the richness of free software in explicitly aesthetic terms, which is one of the reasons that I think Sontag’s work on art criticism is doubly relevant for this book. As far as I know, there is no piece of writing on the aesthetics of software that matches the depth and accuracy of this chapter. To non-programmers and non-geeks, source code often strikes as a functional, mysterious, opaque, and really complicated artifact. Yet for the community of programmers, hackers, and technical users who delve deep into code, it always beholds a dual life: while its fundamental purpose is undeniably functional, it contains, displays, and broadcasts a stunning array of aesthetic elements that not only communicate beauty to the eyes of its creators and

technical admires, but is the glue that bind individuals into a self-reflective community of technical artists.

This chapter discusses the various sources that constitute a unique tradition of computer programming. For example, programming has drawn heavily from a mathematical sensibility that values simplicity, logic, and recursion, yet because of the tremendous constraints programmers work within—hardware specifications and programming language syntax, for example—the artistry of code often emerges most clearly on the edges of creation: that is, as programmers outwit these multiple constraints to arrive at what they collectively value (often after much argumentation) as desirable and beautiful. After various ruminations on the aesthetics and technical requirements of code, Samir and Scott conclude with a tidy and illuminating description of creativity in software. They say: “The act of programming, in its most creative moments, endeavors to meet constraints imposed by nature through the physicality of computation, by the users of the program and their desires for functionality and usability, and by the programming community through the development of shared standards.”

With this basic definition in place they continue to show with more precision how software production is not some simple amalgamation of different styles but stands unique in its own right. And in this regard, the penchant for and importance of collaboration stands out. Of course, all artistic, scientific, and literary endeavors are, to a significant degree, collaborative. The common and often pernicious myth of non-collaboration is sustained and nourished by an ideology of genius, individual authorship, combined with a legal infrastructure of IP. But as *Decoding Liberation* argues, and I think persuasively, “programming is collaborative at a much finer granularity. Modern programming languages and techniques, such as the object oriented

paradigm, are explicitly intended to facilitate this sort of collaboration. In this sense, code is beautiful to the extent it provides an affordance for collaboration or shows its recombinatory potential, by virtue of its clean design, readability, and ingenious solution.” From here we can see most clearly one of the paradoxes that marks the art of programming: only those with the technical skill and know-how can appreciate its beauty, but once inside (this largely boys) club, programming, especially within free software, is far more interactive, collaborative and community-binding and building than many other artistic or scientific endeavors. Further, its collaboration is not secured simply due to technical features but because of the support it receives from the dual pillars of ethics and law. Programmers have created an ethically-dense discourse and set of legal codes that politically support and practically enable this collaboration. Thus, while we can conceptually separate the ethical elements from its aesthetic ones, *Decoding Liberation* reminds us of their close symbiosis.

If much of *Decoding Liberation* fulfills Sontag's call to make art more real for the sake of greater appreciation, much of the book also concerns itself with the possibility of politics and the politics of cooptation. Troubled by the ways in which the mantra of open source may eclipse that of free software, this book takes a direct path, and a clear-cut political position. It demonstrates with great astuteness that if we allow “open source” to become the public face of the free software movement, there is much to lose.

In this regard, I think their work, to some degree, embodies a Nietzschean sensibility and for two reasons. While they certainly bring careful attention to representing the full splendor of free and open source software, they also come to this domain of technological production with a set of values that renders a very clear perspective from which they comprehend its greater

significance. Related to a Nietzschean meta-project, they have a particular dislike for the term “open source” because it conceives itself in apolitical terms. Just as Nietzsche went to great (and really angst-ridden) length to criticize the Enlightenment projects of liberalism and Science for hiding behind a false mantle of universalism and objectivity, Samir and Scott are troubled by open source precisely because it hides its politics by deploying an amoral language of pragmatism.

Samir and Scott provide various analytical excavations that help the reader understand the moral stakes, blinders, and incoherencies involved in choosing the path of open source over free software. While it is impossible here to give any detailed rendition of these numerous and rich excavations, it is important to note that they deploy these critical interventions at three different levels of analysis and using a motley array of philosophical analytics/weapons.

One unit of analysis is that of the developer and the various available licensing schemes to them. Providing a technical (in the philosophical sense) analysis, they largely distinguish between the various moralities embedded in copyleft vs non-copyleft licenses and subject these licenses to a classical ethical analysis; discuss them in light of the division between negative vs positive freedom; and demonstrate how they act constitutionally to demarcate visions of society. (For those that may not know: whereas copyleft requires that future modifications of software fall under the copyleft, thus perpetuating and propagating both the source code and this form of licensing, non-copyleft licensing such as the BSD only stipulates that you must include copyright notices and disclaimers but future fate and state of the software is up to those who modify it). Choosing a non-copyleft license, they conclude exhibits a degree of moral incoherency, and more tangibly, can also erode (or threaten) the commons, as future versions of non-copylefted

software may meet a fate of becoming closed source software.

A second order of analysis concentrates on the types of communities implied in and envisioned by The Open Source Initiative vs those of the Free Software Foundation. For example, the OSI frames the programming community as one that “privileges the technocratic virtue of producing good code,” while the free software foundation “assigns less value to this virtue:” for, “sharing is its primary concern, even at the expense of technical efficiency.” In their reading, “OSI enables a subtle exploitation of the programmers whose code may well be used to further corporate ends and undermine his own community.” Here they demonstrate how even while open source and free software may share the same suite of licenses, they diverge largely through the apparatus of linguistic framing, which as the linguist George Lakoff has recently demonstrated with the example of liberal/democrat and republican/conservative, has enormous power to create distinct moral realities.

Finally, *Decoding Liberation* engages in a much broader set of political assessments. For example, DL discusses how free software can reinvigorate the threatened tradition of openness and peer review in science. They also leave the confines of the academy to conclude in their final chapter on much broader ramifications. In short, in a world permeated by and saturated in software, where technologies literally form the backbone of our government, voting protocols, educational infrastructure, and entertainment, to not choose free software, is as much of a threat to democracy as those threats—corruption and corporate lobbying—we instantly think of as roadblocks to participatory politics.

To end my discussion, I would like to focus on one ironic element made clear to me

while reading this book: Free software is about the freedom to change, read, and alter source code. In recent years, supporters of free software have closely aligned this sensibility to the ideal, so strong in the broader American and Anglo European political imaginary, of free speech. In many respects, we can say that the ideology of open source was born because of a similar freedom to speak and rethink the meaning of free software: to use terminology common in FOSS, open source is a linguistic fork and thus, reflects not only what is possible but even encouraged in the realm of free software. We can't and I don't think we should, stop the process of linguistic reframing—for this sort reinterpretation and dialogical communicative process is integral to ensuring participatory democracies, even while they can threaten it—but what we can do, we must do, and what Samir and Scott have done so well, is to reveal the political consequences that materialize when we choose one path over another, even when the path is largely linguistic in nature. So should always match the freedom to speak with a desire to carefully weigh the consequences of our speech, which is never neutral.

Question:

1. Much of your analysis hinges on closely connecting the OSI with non-copyleft licensing and aligning FSF with copyleft. At one level, I think these connections can be made and are important to make because the similar *effects* of non-copyleft and OSI. You demonstrate how they similarly pose threats to the practices of sharing. However I remain unconvinced that we can really align OSI/Non-copyleft and FSF/Copyleft as closely as you do and this alignment also prevents a critique of free software, which I think, is perhaps implicit in your work. While you are certainly right that the FSF overtly advocates copyleft, the fact remains that as stipulated by the FSD, non-copyleft licenses,

are still entirely kosher. Further, and this is as important, it seems like proponents of Open Source such as OSI and projects that explicitly align themselves with Free Software, like Debian, remain otherwise agnostic on the question of licensing.

Given this context and background, it seems to me that when you unearth the consequences of non-copyleft licensing, it can and perhaps should also be considered not just a critique of open source but to some lesser degree that of free software. For the question remains, if non-copyleft licensing is so problematic, why was it included? Is the FSF caveat of encouraging copyleft licenses enough to get them off the moral hook? Was it a moral oversight? You subject non-copyleft/OSI to an extremely rigorous ethical standard, yet FSF seems to get by relatively easily. Or another way of phrasing it, should the Free Software Foundation excluded copyleft licensing? And if they should have included it, why? Or has the problem of non-copyleft really only materialized in the context of the ideology of open source?