

## **Social Code: "View Source" as a Cultural Strategy**

**4s Conference, October 18, 2003**

**Jenny Cool <jenny@cool.org>**

### Overview

In this paper, I propose the term "view source" to describe an approach to technology that pre-dates open source, free software, the Internet, and the personal computer. My goal in proposing this term is to show that the rhetoric and practices that began in 1998 to be known as "open source" draw on a much broader and older tradition. A tradition not just about access to source and releasing source under a particular kind of license, but one centered in principles of how to develop source in the first place.

In the abstract, my argument is that what people speak of as open source needs to be considered from a wider historical and ethnographic perspective because it is important to be able to separate the particular political-economic movement known as open source, from what I see as a social technology, a mechanism for knowledge production, cultural reproduction and reinvention in a period of rapid social change.

Thus, one of the reasons I find it necessary to speak of "view source" is to distinguish between a set of practices and principles, a set of social tools, and the actors and interests that deploy those tools as cultural strategies. Another value I see in "view source" is that it helps me understand and think about two central propositions in Manuel Castells' characterization of Internet culture.

- (a) "...in the present stage of global diffusion of the Internet, it makes sense to differentiate between the producer/users and the consumer/users of the Internet." (2003:36)
- (b) "The Internet culture is characterized by a four-layer structure: the techno-meritocratic culture, the hacker culture, the virtual communitarian culture, and the entrepreneurial culture." (2003:37)

I will first describe what I mean by view source, then illustrate it with specific historical examples. Finally, I turn briefly to ethnographic examples, returning to Castells' propositions in my discussion of view source.

### **Research Background**

Before setting my case before you, I think it would be prudent to briefly describe the research and experience on which it is based. My fieldwork began in August 1993 when I left Los Angeles and moved to San Francisco to become a participant-observer in the geek culture<sup>1</sup> that begun to flourish there in the early 1990s. As a participant I lived and worked alongside the knowledge workers who

---

<sup>1</sup> I use the term "geek" because that is what these people call themselves.

launched *Wired* magazine, CNet, and developed Apache. Between 1993 and 1999 I worked at Netscape and other Bay Area Internet start-ups and participated in a variety of cultural activities and non-work communities, e.g. SFRaves, Anon Salon, and Burning Man. As an observer I researched telecommunities for the Institute for the Future's (iftf.org) Outlook Project on Intranets and Telecommunity and pursued my own ethnographic work.

In 1999 I left the Bay Area to become a participant-observer of the spread of that culture to Hollywood and the mainstream--I moved to Los Angeles to work as Director of New Media for Disney/ABC Cable Networks. Since 2002 I've been at UC Irvine as a lecturer in Information and Computer Science and as an Internet developer for the Irvine Libraries and Composition program.

This Fall I returned to the University of Southern California and to the doctoral program in Anthropology I left in 1993 to pursue my initial fieldwork in San Francisco. The relevance of this background is that the description and analysis in this paper draw on these ten years of participant observation. This consideration of view source represents my initial effort to frame my fieldwork and distill from it a dissertation.

## **View Source**

The emergence of the World Wide Web and Internet as mass social phenomena is usually traced back to NCSA's release of Mosaic Version 1.0 in April 1993, and the release of Mosaic Netscape in October 1994. One of the interesting things about Mosaic is that it included under the "View" menu the ability for any reader of a Web page to see the HTML source for the page, displayed in a separate browser window.

The ability for anyone with read-only access to a page to display source for that page in the browser, see how it was made, and then build a page of his/her own, was an important vector of dissemination by which the Internet became a mass social phenomena in the mid-late 1990s. The term "view source" derives from this browser feature, but the way of thinking I seek to evoke with this term goes back further. It goes back to the development of the Internet and the Unix operating system, starting in the late 1960s, and to the personal computer revolution that took off in the mid-1970s, and before that to academia and the sciences<sup>2</sup>.

The idea that readers of Web pages would want to view the source, and that this was a basic feature a browser should offer—a browser, not an editor, or an authoring tool— bespeaks a world in which:

- Users are assumed to be active, engaged in valid pursuits of their own; that is, there exists no deep dichotomy of producer/users and consumer/users;
- Data is always coupled with metadata, media is always layered

---

<sup>2</sup> Castells certainly traces the Internets "ideology of freedom" to roots in academic science. Please let me know if you have knowledge of this history or can suggest any leads.

- Self-description and self-documentation are core principles; and
- Revision is a core disciplinary practice.

In such a world, the dominant metaphor for information technology is communication, not computation, not simulation. Every layer, whether physical or digital, is about connecting to—communicating with—layers above and below.

In such a world clarity, visibility, transparency, and documentation are core principles that even a humble web page must embody. Such a world presupposes that users, administrators, creators and audiences aren't neatly separate. They are overlapping, and shifting roles so information about how a thing is made or works should be contained in the thing itself, ideally in human-readable form. In such a world choices are pushed as far towards the user as possible, principles of built-in flexibility and visibility are core.

It is this world which has a specific history and geography that I seek to evoke with the term "view source."

### **View Source in Unix and Internet Culture**

Unix culture is a key part of both the open source movement and Internet culture as a whole. This can be seen in official histories of open source, for example the one on the Open Source Initiative web site ([www.opensource.org](http://www.opensource.org)) which begins:

"The prehistory of the Open Source Initiative includes the entire history of Unix, Internet free software, and the hacker culture."

(<http://www.opensource.org/docs/history.php>)

Both the distinctiveness and broader social importance of Unix can be see in Eric Raymond's introduction to *The Art of Unix Programming* (2003), where he writes:

Unix has a culture; it has a distinctive art of programming; and it carries with it a powerful design philosophy....A very few software technologies have proved durable enough to evolve strong technical cultures, distinctive arts, and an associated design philosophy transmitted across generations of engineers.

The Unix culture is one of these. The Internet culture is another — or, in the twenty-first century, arguably the same one. The two have grown increasingly difficult to separate since the early 1980s..."

(<http://www.catb.org/~esr/writings/taoup/html/ch01s01.html>)

The aspects of Unix most relevant to this discussion of view source are:

- First, in Unix everything is a file, and
- Second, both technically, as an operating system, and socially, as a culture, Unix is communications-centric.

"Everything is a file" is something of a mantra in Unix circles. It is one of the first things said whenever people teach about Unix. What it means is that in Unix-style operating systems everything—from plain files like text and graphics documents, to directories, programs (executable binaries), system commands and device drivers—*everything* is a file. And file names can be long and descriptive. Together these properties reflect an operating system designed for communication from its lowest level. Because everything is a file, everything has a name and most things have a standard location. This creates a kind of openness, transparency, and self-commented nature that runs as deep in the culture as it does in the OS. In the beginning was the file and the file was data and metadata.

Online and in print are many Unix histories, Unix family trees, summaries of Unix tenets, and paeans to the Unix philosophy and culture. Most talk about Unix as a design philosophy, in which "small is beautiful," every program should do one thing well, and where modularity, interoperability, portability, extensibility, simplicity and clarity are sacred principles. As Doug McIlroy, one of the founders of the Unix tradition, put it:

"This is the Unix philosophy: Write programs that do one thing and do it well. Write programs to work together. Write programs to handle text streams, because that is a *universal interface*." (emphasis added)

Doug McIlroy

(quoted in Salus, 1994, the Wikipedia, and many other places online)

One of the innovative things about Unix is that it was the first operating system not written in an assembly language, but written instead in the C programming language, which operates at a level just high enough to be hardware-independent, and is human-readable.

Though Ken Thompson wrote the first version of Unix in 1969, it wasn't until he teamed up with Dennis Ritchie in 1973-74 and they rewrote the Unix kernel in C, that Unix began to spread among universities and small businesses. Until this time, the prevailing wisdom was that operating systems should be written in assembly, "closer to the metal", to effectively access hardware. But Unix shifts the emphasis to communication, rather than access or control. It operates both technically and culturally as a communications-centered system. As Eric Raymond explains the "Rule of Clarity", in *The Art of Unix Programming*:

"write programs as if the most important communication they do is not to the computer that executes them but to the human beings who will read and maintain the source code in the future (including yourself)."

(<http://www.catb.org/~esr/writings/taoup/html/ch01s06.html#id2877610>)

Write programs that can send and receive input from other programs, write so that other programmers can understand what you're doing, do things in simple, predictable, and transparent ways. Every one of these principles connects in to

the dominant metaphor of communication. Communication between programs present and future, between programmers present and future, between programs and users, person to person, person to machine. Communication is the governing paradigm on every level and this lies at the heart of the cultural strategy I refer to as "view source."

In addition to Unix culture, this communications-centric approach to technology can be seen in the groups and processes that gave rise to the Internet, for example in the workings of The Internet Engineering Task Force, or IETF. Aside from TCP/IP, all the basic technology for the Internet was developed via the IETF.

### **View Source, the IETF and the RFC Process**

In the 1980s ARPA began to devolve governance and authority for the Internet, first to the academic and institutional experts who created ARPANET, and later to membership based technical organizations, such as the Internet Engineering Task Force (IETF).

The IETF is noteworthy, not simply for the protocols and standards it developed, but also for its open working group structure and RFC process. These social technologies are part of the phenomena I characterize as view source. Within the IETF, the wider community it serves, and among those who study Internet history and culture, these social technologies are noted for their openness and efficiency in producing working standards.

Though RFC once stood for "request for comment" and this etymology is helpful in understanding view source, as longtime IETF participant Scott Bradner explains: "since documents published as RFCs have generally gone through an extensive review process before publication, RFC is now best understood to mean 'RFC'." (Bradner: 50)

Together RFCs and working groups are the core of the IETF's success and influence as an organization. The organization consists of working groups organized into areas each managed by one or two Area Directors, currently there are 130 official working groups organized into 8 areas. Though the IETF has a Steering Group (IESG) composed of Area Directors and the IETF chair, this group rarely creates a working group on their own. Instead, working groups are proposed to Area Directors by small groups of people. Though working groups communicate by email and meet several times a year, their most significant activity is the publication of RFCs.

RFCs are public documents, freely available on the Internet, and republishable in their entirety by anyone. There are two kinds of RFC, standards track and non-standards track. Standards track documents propose Internet standards and generally originate in working groups, while "non-standards track RFCs can be classified as Informational, Experimental, or Historic" and originate from "working group activity or from individuals." (Bradner: 49-50). These "tracks" refer to the process by which proposals from small groups and individuals are first published

as Internet Drafts, become Proposed Standards, then Draft Standards and eventually achieve the status of Internet Standard.

Each time an RFC moves from one stage of the process to the next, an announcement is made, first to the working group, later to the entire IETF membership, and people are invited to comment on the RFC during a 2-4 week "last call" period. For a Proposed Standard to "be considered for Draft Standard status" there must be at least two "independent, interoperable implementations of the proposal, " all the separate features of the protocol must be multiply implemented, and clear documentation must be published. (Bradner: 51) These requirements are characteristic of view source: technical protocols (data) are documented in human-readable form (metadata), and refined via a feedback process of commentary and revision.

As Manuel Castells has observed, working groups and RFCs are legacies of ARPA's Network Working Group (NWG) which "produced technical standards by consensus, on the basis of request for comment (RFC) documents" from the 1960s to the early 1970s. Castells explains, this "set the tone for future coordination tasks in the Internet: membership based on technical expertise, consultation with the Internet community, consensus based decision-making." (Castells: 29)

Though IETF practices of "open standards, open documents, and open source" are relevant to my illustration of "view source" for their openness, it is important to recognize that active participation in the IETF requires significant technical knowledge as well as interest, time and resources to devote on a volunteer basis. These constraints serve as strong countervailing checks on openness and diversity in the process. Within the IETF there is no deep dichotomy of producer/users and consumer/users, but rather a commonality that supports the building of social consensus, protocols and "view source" technologies.

## View Source and the Personal Computer Revolution

One of my arguments in this paper is that view source is part of a longer tradition, one that goes back further than Linux and open source, to the development of Unix, the Internet, and the personal computer. Thus, before I turn to ethnographic examples, I will briefly touch on this last historical example of personal computing to further illustrate view source.

The personal computer as a mass social phenomena was the work, not of corporate electronics and mainframe giants, nor of the government or military, but of a group of unlikely innovators and entrepreneurs—scientists, electronics hobbyists, phone phreaks, and hippies, working together informally. In *Fire in the Valley: the making of the personal computer*, Paul Freiberger and Michael Swaine give a comprehensive history of the people who developed personal computing. Their account has strong resonance, both with open source and the broader view source perspective I describe. They write:

"This open source idea, what John Draper had called the Woz Principle, is really just an elaboration of the sharing of ideas that Gary Kildall had enjoyed in academia, and that scientifically trained people such as [Doug] Engelbart understood as the key to scientific advance. (Freiberger and Swaine: 423)

At the heart of the pursuit of personal computers with intuitive, graphical and multimedia interfaces lies a presumption that users are active, engaged in valid pursuits of their own. As Freiberger and Swaine describe many PC pioneers had goals of using computers to augment the human intellect, chiefly by harnessing its collective power. From the announcement of the Altair computer in 1975 to the launch of the Apple Macintosh, in 1984, development of the PC was inspired by a variety of communitarian, idealistic and even utopian goals. These goals were sometimes articulated in terms of individuals producing knowledge cooperatively in a massively parallel process, and sometimes spoken of more mundanely, as groupware and shared knowledge bases.

One the most astute observations Freiberger and Swaine make, is that despite their success in bringing computing in to the mainstream, many of the more lofty aims of the innovators and pioneers of personal computing, remain unrealized. As they put it:

"The personal computer has delivered a lot of power to the individual, but hasn't done very much to help people work together in new ways."  
(Freiberger and Swaine: 439)

As always, there was a gap between ideal visions and their realization. One of the fault lines of this gap, as I see it, is the dichotomy between producer/users and consumer/users that deepened as both the code and social technologies that produced Unix, the PC, and the Internet spread to wider public and commercial realms. In defining view source at the outset of this paper, I described it as a world in which:

- Users are assumed to be active, engaged in valid pursuits of their own; that is, there exists no deep dichotomy of producer/users and consumer/users;
- Data is always coupled with metadata, media is always layered
- Self-description and self-documentation are core principles; and
- Feedback and revision are core practices.

The reason I use the construction "a world in which" is that these are systems building and communications principles that have informed the production of knowledge technologies, but they are not *necessarily* tied to a particular group, culture, or ideology. In practice, these principles have been singularly efficient for developing better software. By treating the user as part of the system and assuming continual revision, rather than one time production, these principles structure software development and serve as a mechanism of cultural reproduction among those who do this type of work. For me *view source* is a useful term because it enables me to distinguish a cultural mechanism for the production of knowledge technology, from the specific cultures and individuals who have used that mechanism historically, and continue to use it today.

I also find that view source is useful to thinking about Castells' proposition that:

"The Internet culture is characterized by a four-layer structure: the techno-meritocratic culture, the hacker culture, the virtual communitarian culture, and the entrepreneurial culture." (Castells: 37)

When I first encountered this description, I was puzzled about whether Castells was describing a synchronous culture composed of four simultaneous elements (as words like "layer" and "structure" suggest), or whether he was describing the historical layers of cultures that have contributed to Internet culture. Each of these cultures exists outside the Internet and, however apt it is to cite their convergence in the Internet as a mass phenomenon, I am not certain it is appropriate to speak of them as constituting a singular *culture*, rather than as contributing and contesting *cultures*.

View source is common to all four cultures Castells names and I see it as integral to Internet cultures in general. However, the first principle of user/producer propinquity, is often compromised in the entrepreneurial culture, which tends toward a dichotomy of producer/users and consumer/users, and frequently assumes consumer passivity. The techno-meritocratic, hacker and virtual communitarian cultures, on the other hand, have tended to use view source strategies to built tools for themselves and others like themselves. In doing so they have exploited a special case of feedback in system design—the case where there is an identity of producers and users, or at least close community and continuous communication.

## **Ethnographic Examples**

In the final part of this paper I turn to two short ethnographic examples. With the first I seek to demonstrate how widespread the view source approach has become, and with the second return to my discussion of Castell's distinction between producer/users and consumer/users.

### **Example 1: The Right to Reverse Engineer**

The attitude that if a thing has been done anyone with the technical skill should be able to take it apart and learn how it was done is so entrenched that I speak of it as the "right to reverse engineer." It is not only open source developers who speak as if this were a right. This aspect of view source has gone mainstream. It is widespread among my computer science students, most of whom are life long Microsoft users. Every quarter I see papers where students explain that the technical features they propose will not be difficult or costly to implement, because someone else has already implemented them in another product. For example, and I quote:

"These changes are fairly simple to make and have low cost because the technology already exists as other peer to peer systems have already incorporated some of the features into their programs." (Student Paper, 2003)

This attitude applies regardless of whether a particular technology is open source or not and that is why I see it as an expected right to reverse engineer. Edward Felten, the Princeton Computer Science professor involved in a legal battle over academic publication of techniques for breaking encryption schemes, writes about the "freedom to tinker" as "your freedom to understand, discuss, repair, and modify the technological devices you own." (<http://www.freedom-to-tinker.com/>)

### **Example 2: Calliope Project**

In 2003 I began studying the Calliope project, a volunteer group of open source programmers in the Bay Area. The goal of Calliope is to "develop an online community forum that allows users to post messages, pictures, share files, and other functions with protections for the community including strong privacy protections and no commercial advertising." Calliope began as a project of the Online Policy Group (OPG), "a nonprofit organization dedicated to online policy research, outreach, and action on issues such as access, privacy, digital defamation, and the digital divide." Their motto is "One Internet with Equal Access for All."

Though there isn't space in this forum for detailed description of Calliope, I will highlight some of the key themes of my view source argument from a single interview with Alex, one of the developers who took on the project for OPG.

An articulate, white male in his early 30s, Alex answered my question about the relation of his work to open source as a whole, in the form of four theses

that resonate with my discussion of view source. Let me excerpt from two of them:

- "Thesis 1: View source, the idea that if you can see the source code, you can improve it and manipulate it to meet your needs. In general, the myth is true. I use it all the time, like when I'm using source code libraries in my own programming work. Jakarta commons, programming utilities, tools that do common things that should be in the core Java API, but aren't for some reason or another...but one thing the myth seems to **miss** is that that works for code at the library level or the component level, but it doesn't seem to work so well for code at the product level."

What Alex is saying is that view source as a strategy for developing software works well when producers and users of the software are the same people, as is the case with code libraries and programming tools which are used by software developers. However, when developing products for consumer/users view source does not necessarily provide great advantage. View source as a cultural strategy depends on much closer ties of identity and/or commonality between producers and users, than mass-market capitalist production generally affords. Or, to put it another way, it remains to be seen whether the strategy will scale as the Internet culture Castells speaks of grows to the point where it "makes sense to differentiate between the producer/users and the consumer/users of the Internet."(Castells:36)

Alex's fourth thesis also points to this dichotomy between producer/users and consumer/user, but focuses on revision as a core disciplinary practice.

- "Thesis 4 is feedback, and that's actually the most important thesis...Being developed by the users is a special case of a much larger issue which is feedback, which I've come to realize ideologically or whatever...I've come to realize that feedback is the most important force in the universe. The most important force for good. Because everything sucks...right, anything you do sucks on the first try, no matter what it is. It always sucks. So the trick is to rev it again and again and again. And you don't just rev it based on what you think and what you're speculating, and what you alone think is going to improve it. You need to rev it based on the maximum amount of feedback based on the maximum amount of people. And you keep reving and keep reving..."

The development process Alex describes here encapsulates view source, particularly the centrality of feedback and revision. It also shows that, even though Alex is volunteering his time on an open source community project, neither politics, nor software licenses, are the focus. The focus instead is on the practical strategies for developing software that I've described throughout this paper as view source. Such strategies do not have a fixed or specific politics or philosophy, but they are deeply cultural.

## Conclusions

Diverse groups have taken a view source approach to technology. The practices and rhetoric of openness that Castells calls the Internet's "ideology of freedom" are now common, not only among a variety of Linux, Free BSD and other open source software developers, but also at large corporations, government and academic institutions. It is crucial not to equate this shared set of practices with a consistent set of shared values. As Castells puts it: "freedom has many uses" (37).

As an approach to software development, the view source tradition has historically proved exceptionally efficient because it is centered in principles of how to develop source in the first place. Yet these principles were forged at a time when users, administrators, creators and audiences weren't neatly separate: a time before Castells differentiation of producer/users and consumer/users became necessary. While many contemporary open source projects aim to reach a consumer audience, it remains to be seen whether techniques developed among groups of peer users, who share a good deal of technical and social knowledge, will work where the dichotomy between producers and consumers becomes much greater.

## Sources

Bradner, Scott

1999. "The Internet Engineering Task Force" in *Open Sources: Voices from the Open Source Revolution*. Sebastopol: O'Reilly.

Castells, Manuel

2003. *The Internet Galaxy: Reflections on the Internet, Business, and Society*. Oxford: Oxford Press.

Felten, Edward

2003. "Freedom to Tinker: Reconciling Cultural and Technical Creativity" Copyright and the Networked Computer Congress, Washington D.C.

<http://dc-mrg.english.ucsb.edu/conference/cncsc/index.html>

Freedom to Tinker Web log: <http://www.freedom-to-tinker.com/>

Freiberger, Paul and Michael Swaine

2000. *Fire in the Valley: The Making of The Personal Computer*. New York: McGraw-Hill.

Open Source Initiative

2003. "History of the OSI."

<http://www.opensource.org/docs/history.php> (last viewed 11/17/2003)

Raymond, Eric R.

2003. *The Art of Unix Programming*. Boston: Addison-Wesley.

<http://www.catb.org/~esr/writings/taoup/html/> (last viewed 11/17/2003)

Salus, Peter H.

1994. *A Quarter-Century of Unix*. Boston: Addison-Wesley.

Scoville, Thomas

1998 "The Elements of Style: UNIX as Literature" Miller Freeman.

<http://www.rap.ucar.edu/staff/tres/elements.html> (last viewed 11/17/2003)

Wikipedia

2003. Entry for "Unix Philosophy"

[http://en2.wikipedia.org/wiki/Unix\\_philosophy](http://en2.wikipedia.org/wiki/Unix_philosophy) (last viewed 11/17/2003)