# ***HAU* TO DO THINGS WITH WORDS**

Christopher M. Kelty

Assistant Professor, Department of Anthropology

Rice University, 6100 Main St., Houston TX, 77005

## **Revision History**

Originally Written November 2000.

Substantially Revised April 2001.

Slight Revisions, released under CCPL December 2002

*Hau to do things with Words,* Christopher Kelty, Rice University

## Introduction

…For one who says 'promising is not merely a matter of uttering words! It is an inward and spiritual act!' is apt to appear as a solid moralist standing out against a generation of superficial theorizers: we see him as he sees himself, surveying the invisible depths of ethical space, with all the distinction of a specialist in the *sui generis*. Yet he provides Hippolytus with a let-out, the bigamist with an excuse for his 'I do' and the welsher with a defense for his 'I bet'. Accuracy and morality alike are on the side of the plain saying that *our word is our bond.*.

--J.L. Austin, *How to do things with words*, p. 10

Since 1998, the terms "Free Software" and "Open Source"[1] have become a common feature of talk about the software industry, the internet, and the political and technical structure of society. An admirable range of lawyers, activists, academics, and engineers have become part of a discussion once confined solely to hackers, geeks, and a handful of academics in specialized fields dependent on computing and networking. What was once regarded as a hobby has become a central feature of discussions about intellectual property law, about commercial software contracts, about the openness or modifiability of software, about the availability of scientific data, about the nature of freedom of speech on the internet. Free Software has brought these issues together in a manner that indicates that the divisions people are used to—law, art, technology, ethics, science etc.—can't capture the problem. They are divisions of a critical discourse inadequate to the technical fact of Free Software.

This article is a general introduction to these interrelated aspects of the phenomena of Free Software. It is not a *critique* of Free

Software, of hackers, of intellectual property, or of any "culture" or "cultural practice" of software programming or entrepreneurial capitalism, though it does attempt to put certain of these issues more clearly and correctly than they have been put to date. There is little to be gained from an overly detailed or aggressive critique of hackers, engineers, entrepreneurs or the media, because the point I want to make is that *Free Software is itself a species of critique*, leveled at a particular configuration of business practices, and manipulating property and contract law to these ends. Through its technical and legal practice, it explicitly changes the political-economic structure of society.

This powerful notion is far from having gone unnoticed. There are several sustained attempts to explain the genesis and structure of this state of affairs, mostly by individuals who also write and/or promote Free Software. These explanations—especially Eric Raymond's, which I will survey in the second third of this paper—are explicitly offered as "scientific" anthropological or economic explanations, even as they come from individuals whom an "anthropologist of cyberculture" might be tempted to label "indigenous" to the hacker culture. While these "indigenous" explanations have fallen back on a sort of vulgar anthropological explanation—a mixture of common-sense economics, natural selection, and popular culture-influenced beliefs about the cultural and technical evolution of societies—they are nonetheless widely read and cited by academic anthropologists, economists, lawyers, and sociologists who have chosen to study Free Software. This makes the fiction of an indigenous explanation pedantic, at best; at worst, it allows the scholar to actually miss the *importance* of the development of Free Software.

Instead, it is probably more accurate, and less disingenuous, to insist that I am competing—or collaborating—with my "informants" to offer a better, more complex, perhaps even more *scientific* explanation of Free Software. This article doesn't attack Free Software, nor does it offer practical suggestion for its improvement. In fact, it is safe to say that I am already in near complete agreement with the current aims of Free Software and its explainers—I think it is practically and ethically essential to both practice and promote it. The goals of Freeing software—the creation and maintenance of a public domain, the enlargement of the sphere of actual economic competition in software, the protection of rights to privacy and control over information—these are things that need both promotion and justification in specific contexts, and I consider this work to be in sympathy with those goals.

However, there are a set of debates—as it were, indigenous to anthropology—which raise a very different set of issues and which are related to the topic of Free Software in complex ways. Free Software and Open Source are often promoted and explained as "gift economies" by both advocates and observers alike.  This usage, which derives primarily from writings by Howard Rheingold and Eric Raymond, is a common sense consensus of the notion of a gift economy:  that it is a closed, non-monetary sphere of exchange based on an alternate currency of trust—reputation.  None of the people who explain Free Software in this way use the work of Bronislaw Malinowski or Marcel Mauss, or the subsequent  tradition of social theory and investigation of exchange.   This is perhaps because the common sense notion is good enough for the purposes of advocacy.  But Free Software, as a phenomenon which Marcel Mauss might have called a "total social fact,"  actually offers anthropology a very specific object with which to re-read this tradition of studies of exchange.  Therefore, I do not intent to investigate Free Software by using Marcel Mauss, but exactly the opposite: to investigate Marcel Mauss with Free Software.

Nonetheless, this is impossible without first introducing Free Software and attempting to explain as clearly as possible, what it is (Part 1).  This is followed by an extensive introduction to Eric Raymond's explanation of Free Software—or as he prefers to call it, Open Source software development.  Raymond adopts the identity of an anthropologist to offer this explanation, and so I intend to treat it as a part of anthropology—or of the social sciences more generally—even though it might seem unfair to hold Raymond to these standards (Part 2). Finally, I offer a reading of Marcel Mauss in which Free Software is employed to help illuminate Mauss' theories of gift exchange—and as I refer to them—"the structures of memory and expectation" that are involved in that theory (Part 3).

## Part 1: How to Free your Software

### A four step approach

There are several steps to freeing software.

Step 1: Get computer, write software. The first step is the hardest: it requires an extensive knowledge of the world of computer operating systems, the functioning of computers, the various possible programming languages, networks, protocols, development software – and, most importantly, a zen-like attitude towards the proper placement of special characters like parentheses or hash marks. It requires no math, no physics, and to write it you do not have to be “good with machines”. Nonetheless, the first step might take a few years.

Step 2: Make your “source code” freely available to anyone. “Source code” is a shorthand for the “human readable version” of a piece of software – your definition of human may vary. Source code, with all of its human-readable instructions, variables, parameters, comments, and carefully placed curly brackets is processed by a compiler which turns it into "object code": a binary, executable program that is specific to the architecture of the chip and the machine that will run it. This is an adequate explanation, though it is important to note that the distinction between source code and object code is not firm.[2] Likewise, the term "freely available to anyone" is flexible. In this particular context it means that Free Software is anonymously downloadable from the internet or available for a small fee on diskette, CD-rom, or any other medium. In a perhaps more trivial sense, freely available also means "not kept secret"—secret software cannot be Free.

Step 3: Copyright your source code. Assuming that you can get your code to work – which is not trivial – the next step in creating Free Software is to copyright it.[3] In the world of software production there is no more powerful institution than intellectual property, and it is arguably as important to Free Software as it is to proprietary software (Coombe, 1998; Boyle, 1996). Copyrighting the source code is a necessary, but not a sufficient condition for software to be Free.

Step 3a: Pause to consider the allocation of functions among patent, copyright, and trademark for a moment. Patents are generally understood as the protection of the "idea" of a technology; when patenting software, applicants generally avoid submitting the *actual complete* source code in the patent application, offering instead a *representation* of the idea which the code expresses.[4] Copyright is more straightforward, and consists of asserting a property right over an original text by simply marking it with a ©. Thus when one copyrights software, one asserts rights to the *actual* technology, not to a *representation* of its idea. As with a novel, copyright covers the actual distribution and order of text on the pages – and sometimes extends to something less  exact, as in the case of Apple's Graphical User Interface.[5] A different version of that "idea" can be copyrighted in its own right, just as a rewriting of *Macbeth* can. Trademark, finally, is an even stranger beast, intended to protect the authenticity of a work. Since the nineteen-eighties – when it became customary to add  the value of a brand identity to a corporate balance sheet,[6] trademark has ceased to act as a failsafe against "consumer  confusion" and has become a tool for the protection of assets.

Step 4: Add some comment code. Comment code is not source code; when a user compiles a program, the compiler compiles the source code and ignores the comment code. Some people—for example, computer science professors teaching undergraduates—insist that comment code is *essential* because it is the medium by which one explains (for example in English) to another human what the software should accomplish. Comment code can be just as opaque as "real" source code, but very few people would argue that comment code is technically necessary. Source code lacking comment code will still technically work, but everything depends on your definition of technical—the machine may understand it, but the human may not.[7]

In the case of Free Software, however, the particular piece of comment code to be added is anything but non-technical: it is a legally binding contract license which allows the user to do a specified set of things with the source code.[8] There are many variations of this license, but they all derive from an ur-license written by the Free Software Foundation  called the General Public License or GPL . This license says: copy and distribute this code as much as you like, but only under the condition that you re-release anything you make with it or derive from it with the above copyright and contract attached. Software licenses are exceedingly common today. Almost all proprietary software includes a license called an End-User License Agreement (EULA) known in the legal profession as a "click-wrap" or "shrink-wrap" license. These licenses are agreed to merely by installing or using the software. Most EULAs govern what a user can or cannot do with a piece of software. Copying, modification, transfer without license, or installation on more than one machine are often expressly prohibited. The GPL functions the same way, but it grants the user the opposite rights: rights to modify, distribute, change or install on as many machines as needed. GPLs are not signed by using the software, they are only activated when the software is re-distributed (i.e. copied and offered to someone else either freely or for a price).

Your software is now Free. The process is commonly called “copy-lefting” the code[9].

## Legal Hacking

It is only the combination of copyright and contract law in this peculiar and clever manner that allows software to be free. Free Software, as it originated in with the Free Software Foundation, is explicitly opposed to use of intellectual property rights to keep software source code from circulating. It therefore uses contracts like the GPL to guarantee that the holders of the intellectual property rights (such as, for instance, The Free Software Foundation, which holds a large number of the copyrights on existing Free Software) enter into an equal agreement with the subsequent user or purchaser of the software. Some explanation of both of these legal regimes will clarify this situation.

On the one hand, intellectual property law organizes one entity’s rights over a particular thing, vis-à-vis any other (potential) person. As is clear from debates in legal theory and practice, intellectual property is not just a conceit built on the supposedly obvious notion of exclusively possessing tangible things. As Horowitz (1992), Sklar (1988), and Commons (1968) variously argue, property in North

Atlantic Law is about defining the allocation and relative priority of a "bundle of rights." The legal structure that organizes the allocation of these rights should not be confused with the evaluation of the objects themselves, which requires particular institutions such as the United States Patent and Trademark Office. All too often, the fact that something is patented or copyrighted is taken to imply that it is useful, non-obvious, accurate, workable, efficient, or even true. While these criteria may be important for the decision to grant a patent, the patent itself only makes the object property; it grants the designated inventor a limited monopoly on the sale of that item. Therefore, information and land, in this sense, cannot be usefully distinguished with respect to tangibility: both are simply useful legal fictions. Though we may be tempted to ask "how did information become property?" the question might be more usefully phrased: "how did property become information?"

On the other hand, contract law governs two separate persons (individual or corporate) rights to a third thing or person.[10] Like property law it concerns the allocation of rights, governing conflicting claims by rival individuals to a third thing or person. However, it is activated only in the case of a violation of the terms of the contract. Contracts are definitions of rules for the parties involved, for the duration of an agreement. Only when such rules are violated must some higher authority (the court, for example) step in as adjudicator.

In the case of Free Software, whatever a user decides to do with the source code (compile it into a binary executable, change it, add to it, etc.), the contract guarantees that what she has been given can be used for any purpose, and furthermore, that it will *generate further giving,* by requiring her and each subsequent user to agree to the same set of requirements. The contract assures that the subsequently modified code cannot be re-appropriated by anyone, even the original copyright holder (unless the contract is ruled invalid). Hence, anyone can take, give, see, learn from, install, use, and *modify* a copy-lefted piece of software.[11]

As various people have observed (see especially Lessig, 2000), this very clever use of the laws of property and contract makes the legal system a kind of giant operating system; using the system in this manner constitutes a "legal hack."[12] While it is technically superfluous to the software, the contract – contained within the copyrighted (and hence appropriated) source code of the program – is legally binding. It guarantees that anyone can do anything they want with the software, *except change this license*. They can use it, not use it, modify it, not modify it, and they can even sell it to a third person, provided that this third person is *willing* to pay, either for the software or for associated services.

For someone who can't make heads or tails of software, the license may indeed seem superfluous: why would an ordinary user care whether or not the source code is visible or modifiable?[13] The answer is simple: because all software exists in an inherently heterogeneous, evolving environment of other software, hardware, devices firmware, operating systems, networking protocols, application interfaces, windowing environments, etc.. Software needs to be flexible. It must work not only in a particular setting, but it must continue to work as other aspects of the system change. In the case of proprietary software this creates an impossible situation – the user must rely on the corporation that *owns* the software to change or fix it and, regardless of her skill, is not *allowed* nor *enabled* to do so herself. Relying on a software corporation, as most users know, is at best a very uncertain proposition.

Free Software has grown up with the internet. Most of it is part of long and rich history of university-funded software and protocols that are open and freely available, primarily because they are created and funded by government and universities. Much of Free Software, such as the operating system GNU/Linux, is explicitly built on work done since the early seventies at DARPA, AT&T Bell Labs, the US Government, MIT, UC Berkeley, Carnegie Mellon, and the National Center for Supercomputing Applications. However Free Software is not in the "Public Domain" as most scientific data is presumed to be, or as are the basic protocols and standards of the internet and the web.[14] Rather, and as it may seem contradictorily, Free Software is *protected intellectual property that* *anyone can use*.[15] It is a form of commercially contracted openness-through-privatization and contributes to the creation of a commercially and legally legitimate self-reproducing public domain of owned property-- property anyone can use simply by agreeing to grant that right to any subsequent contractee.

## Part 2: Anthropologist vs. Anthropology or, How to explain Free Software

I hate traveling and explorers.

--Claude Levi-Strauss, *Tristes Tropiques, p. 1*

Everybody has an explanation of Free Software. Some rely on the anecdotal, the personal, and the certainty afforded by experience and the power of hindsight; some, not always the younger or less experienced, rely on the rhetoric and language of science. Occasionally both are combined in a single person, as we will below see with Eric Raymond. Some people want to explain the legal or technical issues in detail, others want to understand something larger, something—"anthropological."

Perhaps as a result, the most popular autobiographical and scientific identity for hackers, geeks, computer engineers and software developers to adopt when they aren't working, is that of the anthropologist. Perhaps it is because the definition of anthropologist is fuzzy enough to include both authentic experience wrenched from the tedium of everyday coding *and* speculative scientific derring-do wrenched from actual biological or economic research. Over the years, there have been several such adoptions. It began in the pre-*Wired* era with Carl Malamud's *Exploring the Internet: A Technical Travelogue* (Malamud 1992) in which Malamud scoured the earth, correlating the minimal connections of the late 1980s internet with the peculiar cultural habits of its denizens—a parodic compendium that thankfully swerved past behavioral theorizing. In 1996, science fiction writer Neal Stephenson was paid by *Wired* to jaunt around the globe carrying a handheld GPS and a notebook following the laying of the FLAG undersea cable, and filling pages with data haven stories, historical connections and human behavior seen through a fiber-optic darkly (Stephenson 1996). It would eventually become *Cryptonomicon* (Stephenson 1999b) no less an anthropological theory for being a novel. Around 1993, Howard Rheingold, most famous for his book The *Virtual Community: Homesteading on the Electronic Frontier*,[16] used the notion of a "gift culture" to create an internet-geek consensus of the “virtual community” as a kind of Romantic-Rousseauist space—a conservative post-modernism in which the scientific and technical advances of the 20th century usher humanity into a transcendent realm free of corruption, privilege, or elitism. Today, critiques of such "strategic optimism"[17] are as plentiful as the visions themselves—now discarded on the remnants of a dotcom bubble that lasted nearly three years.

It is Eric Raymond's work (1996, 1997, 1998, 199a/b), done under the explicit mantle of anthropologist[18], and expanding the metaphors of Rheingold's meditations, that has recently proved to be most interesting—and despite itself, the least anthropological.[19] Raymond's work has the virtue—or vice—of transforming excellent ethnographic observations into un-tethered biological and economic speculation. The particular nature of his speculation is interesting in itself, as it points to one strand of techno-libertarian thinking that dominates discussions of the internet—in particular that the realm of law and legal regulation are at once unnecessary and detrimental to the proper evolutionary development of technology. However, since Free Software exists only because it manipulates law and legal structures, the techno-libertarian speculations make Raymond's valuable ethnographic observations more difficult for the average reader to glean. Accordingly, the following sections focus on Raymond's work, the role he has played in the history of Free Software and Open Source Software, his ethnographic observations, and the way he reaches his conclusions by strategically denying the role of law and politics.

Eric Raymond is everyone's leader and no one's hero: despite his extreme political views, his version of Free Software—which he calls

Open Source—has seeped into nearly  everyone else's explanation, in some form or another.  Raymond's public involvement with Free Software is second only to that of Richard Stallman, the head of the Free Software Foundation and Raymond's longtime friend and ideological nemesis. Raymond  is well known for "fetchmail" – a great mail-transfer program he wrote – and "The Jargon File," a collaboratively maintained collection of technical terms and definitions which he published with MIT Press as the second and third editions of "The New Hacker's Dictionary."[20]   He has also written widely on subjects ranging from his leadership in the "Geeks with Guns" movement to installing Linux, to the evolutionary psychology of infidelity.

Raymond is probably most well known for the article "The Cathedral and The Bazaar" (hereafter CatB) (Raymond 1997 and 1999a) which generated massive interest in Free Software and Linux in particular (it was originally presented at a yearly Linux geek gathering called LinuxKongress).  CatB changed the way people – especially people with money – perceived geeks. The article explained the mechanics of Free Software development to a broad audience while at the same time maintaining a technical proficiency that satisfied most hackers—indeed, even gave them a way of explaining it to others.

Since then Raymond has supplemented CatB with two more lively and speculative pieces. In “Homesteading the Noosphere” (hereafter HtN) (Raymond, 1998) he provides an anthropological explanation of why hackers engage in developing free software; the article also contains suggestions for how to judge which software projects are worthy of reputation. “The Magic Cauldron” (Raymond, 1999) suggests various business models for open source software.  It is also in these three articles that Raymond offers his thoughts on the nature of gift cultures, property customs, and informal rules governing collective action amongst what he calls “the Hacker Tribe.”

A little bit of history will help to understand why exactly there are two names for the same thing—Free Software and Open Source—before we can proceed to the question of whether or not it matters, why it might, and how it matters to the explanation of what it is.

## From Free Software to Open Source

The story of  the Free Software Foundation (FSF) has been told over and over again, and the stories continue to appear.[21]  It was founded by Richard Stallman in 1983 and  existed in more or less the same small-scale form until about 1991.  Between 1991 and 1997, with the explosion in size and access to the internet, its vision and its software started to reach a much larger audience.  As a sub-cultural phenomena, it was generally ignored or under-reported until about

1998, when something significant happened.

In January 1998, Netscape publicly conceded defeat to Microsoft in the “browser wars.”[22] Netscape was already famous for its startling behavior: giving away Netscape for free and thereby forcing Microsoft to do the same, and holding a media-saturated initial public offering with a business plan that had no clear revenue model. In response to Microsoft's victory in browser space, Netscape decided to up the ante: they would release the source code to Netscape – now pronounced Mozilla – and make it a Free Software browser. The point at which they decided to do this follows closely—or so the anecdotal story goes—on the heels of a marketing meeting where members of management had invited Eric Raymond to come and talk about CatB and the dynamics of Free Software development. Apparently, Raymond convinced them to make the source code available.

Immediately following this, on February 3, 1998, Eric Raymond announced that he will no longer use the name “Free Software,” but instead would call it “Open Source Software.” No one (except the CIA) used the phrase "open source" prior to this date.[23] Raymond's justification was that in order to make a better case to potential business users it was necessary to avoid using the word "free". Apparently the use of the word Free – which was intended to mean Freedom – had baffled businessmen, and had led venture capitalists to assume that Free Software was not a legitimate aspect of the business world but rather a hobby for nerds or, worse, a hotbed of communist organizing.

At this point, Raymond joined with Bruce Perens, a long time Free Software advocate and member of the volunteer organization that created the distribution of Linux known as Debian, to create the Open Source organization. They took a document written by Perens and called the "Debian Social Contract," and converted it with minor changes into the "Open Source Definition." As the Open Source organization, they issued press releases that featured Linus Torvalds and Eric Raymond promoting the “open source” strategy.

Raymond and friends had by 1998 recognized that the *commercial* internet depended – uncharacteristically perhaps – on *Free* Software. System administrators everywhere, even inside Microsoft, were using Free Software like Apache, Perl, Bind, and Linux.

This was largely because the internet was a new, and for most businesses—especially Microsoft—unfamiliar technical space, whereas much of Free Software is actually designed with the internet in mind. Beyond that, most of the people closest to the machines—such as system administrators and networking specialists agreed that Free Software is faster, more stable, more configurable, more fault tolerant, more extensible, cheaper, easier to get almost anywhere in the world, less buggy, comes with a worldwide network of support, and well, it just *works*[24]. Internet pioneers like Amazon and Yahoo would never exist without the work of the Free Software community, and it was clear to Raymond that the time was ripe to do something proactive about it.

For Raymond, this meant something very specific. Hackers should strategically repudiate the name "Free Software," and especially any reference to Stallman's rhetoric of freedom. To Raymond, Richard Stallman represented not freedom or liberty, but communism. Stallman's insistence, for example, on calling corporate intellectual property protection of software "hoarding" was doing more damage than good in terms of Free Software's acceptance amongst businesses. So, riding the rising tide of third wave capitalism and e-everything pre-millenarian madness, Raymond's response was to expunge all reference to freedom, altruism, sharing, or any political justification for using free software. Instead, he suggested, hackers should promulgate a hard-nosed, realist, cost-cutting, free-market business case that free software was simply better—and more economically efficient as a result. Capitalism had triumphed, the future was determined, it was all over but the shouting. No politics, just high quality software – that was the deal.

Raymond's intuition was right. That is to say, "Open Source" did prove to be a better name—from the perspective of popularity if nothing else.  It highlighted the importance of the source code instead of the issue of Freedom.  While Raymond's justifications for the change were somewhat suspect—perhaps tied to the marketing concerns of Netscape—the fact is that the combination of Netscape's announcement, Raymond's article, and the creation of the Open Source organization led to massive, widespread industry interest in Open Source, and eventually, in the spring and summer of 1999, to major media and venture capital attention. Previously little-known companies such as Red Hat, VA Linux, Cygnus, Slackware, and SuSe, that had been providing Free Software support and services to customers suddenly entered media and business consciousness.[25] Over the last two years several large corporations, like IBM, Oracle, and Apple, have decided to support various Open Source projects.

Raymond and Open Source achieved a kind of marketing revolution—indeed Bruce Perens (who subsequently resigned)  now refers to the Open Source organization they founded as "a marketing tool for Free Software."  It is perhaps unclear whether Open Source would have been the most recent "next big thing" if it were still called Free Software, but the fact is that the name, the idea, and some of the money that came as a result, has stuck.  Something forgotten amidst this marketing maelstrom is that  Open Source did actually have a unique, specific, material goal in mind. The Open Source organization and the Open Source Definition decided to change one thing: they would offer a "certification mark," to protect software that is "authentically" open source.[26]

This is somewhat peculiar, even if it seems eminently reasonable at first glance. As we have seen, legally speaking, the only thing "authentic" about Open Source – the only thing that distinguishes it both *legally* and *technically* from proprietary software – is the Free Software license itself. Without it, it is just copyrighted code. Such code *could* be trademarked, as Windows98Ô is because Microsoft, as a legal entity, owns both the copyright and the trademark. Open Source software, on the other hand, may be owned by someone, but the Open Source organization—even as a legal entity—cannot trademark it unless it *is* that owner. So instead, the Open Source organization can only offer a certification mark that represents their guarantee that software so marked is actually Open Source software— i.e. it contains a Free Software license.

Note that the Free Software Foundation and the Open Source Organization recognize more or less the same list of licenses.[27] This means, then, that an open source certification mark can have no other purpose than to certify that the software is in fact *licensed correctly,* which despite Raymond's non-political revolution, is precisely what the

Free Software Foundation always insisted on doing—albeit in a ethical voice, not through the power of trademark law. After all, the one necessary condition for software to be Free is the Free Software license. Vague and indeterminate definitions of "Free Software" or "Open Source" are open to endless debate, but its licenses are exact, fragile, legal instruments that achieve a precise political maneuver within a particular institutional milieu. So in practical terms, there is no difference between calling it Free Software and calling it Open Source software—it's all made of the same stuff. Inasmuch as Open Source is simply a better marketing term for the same stuff, it cannot actually function that way in legal terms. Whereas Pepsi and Coke have tremendous amounts of value tied up in protecting their brand, and in owning the trademark, Open Source is not a for-profit corporate entity which produces an excludable product, and therefore cannot exclude anyone from using the name. This means that Microsoft can legally call their software "open source" if they want, but they won't be *certified* by the Open Source Organization to do so. I belabor this point because it is rare to find a situation where the choice between two equivalent names, and the difference it makes, can be seen so clearly: for those who value the political import of Free Software, the name Open Source mortgages the future of that political goal. For those who value the wide and popular acceptance of Open Source, the name Free

Software sabotages the possibility of including the broad range of pragmatic software users, regardless of political "ideology". It is clear in this case, though, that it is not just what the words mean that matters, but it is also what they *do* that matters—and this is an issue of the techno-legal framework of contemporary society.

There is, however, another part to this story, and another reason why some people use Open Source, and some Free Software. This has less to do with the precision of law, and more to do with several important empirical and historical developments that converge in the 1990s. For Eric Raymond licenses like the GPL, and associated trademark and copyright issues, are a secondary and less important part of the story. The existence of the GPL is a necessary but not a sufficient condition for what Raymond *wants* the new term "Open Source" to mean: *the distributed, cooperative, evolutionary design of high quality software by non-corporate organizations of independent developers.* Open source development, defined thus, is what fascinates Raymond—licenses are important, but they are not the heart of the matter. The heart of the matter is that a bunch of volunteers, with asynchronous access to openly available source code can build a highly complex piece of software in the absence of any explicit corporate management. Raymond proselytizes for a better bug-trap, a new software development model and this is what CatB is all about: make it open source and as a result of the evolutionary distributed dynamics of Open Source, it will simply be better software than if you close it up and let only one company develop it.  And so, you don’t need political justifications to convince people—the  software will sell itself.

Because Raymond had been a fervent supporter of , and long time participant in Free Software and because he is also a committed amateur anthropologist, Raymond has written a very widely read, occasionally convincing set of explanations for how, and even *why*, it works so well.  It is worth revisiting the details of these two articles: the first (CatB) sets out certain questions about the dynamics of cooperation, exchange, and the technical side of software development.  The second (HtN) contains Raymond's anthropological explanation of property customs and reputation allocation for Hacker Tribe that he calls a "gift culture".

## The Cathedral and The Bazaar

In “The Cathedral and the Bazaar” (Raymond, 1997), Raymond proposes  an experiment[28] to prove that the evolutionary dynamics of open source development (the Bazaar) are more efficient than those of the in-house hierarchical software development model (the Cathedral).

The article reviews the development of the Linux kernel started by Linus Torvalds, and proposes the fetchmail mail-transfer agent maintained by Raymond as a confirmation of the dynamics of Open Source. The goal of his paper is to identify the difference between the closed software firm model, and an open, distributed, collaborative model. The difference he identifies is the role of debugging. In most cases of software development, the code is designed, the data structures and flow of the program specified, and then a version built, usually by a small number of people. The next step is debugging and in the Cathedral model, claims Raymond, the same small handful of people are assumed to be the best bug-finders, and therefore only they get to see that code. But the way Linux was developed, the code was always open and anyone could look for bugs at anytime during its development: .

> Linus was aiming to maximize the number of person-hours thrown at debugging and development, even at the possible cost of instability in the code and user-base burnout if any serious bug proved intractable. Linus was behaving as though he believed something like this:
>
> 8. Given a large enough beta-tester and co-developer base, almost every problem will be characterized quickly and the fix obvious to someone.
>
> Or, less formally, "Given enough eyeballs all bugs are shallow." I dub this: "Linus's Law". (Raymond, 1997, Section

4).

Note that Linus's Law is not a law of how cooperation works, it is only a  description of a particular software development technique. That is, it does not focus on *why* people contribute to such projects; rather, it lays out what should be done to achieve this contribution. Two related aspects of the model are important here: 1) users *are* developers and, 2) everyone deserves some *credit* for helping. According to Raymond, if these two conditions are satisfied, the software will not succumb to deep, insidious design flaws but rather become steadily more robust, fault tolerant, flexible, and useful to a wide range of people.

Recognizing that users are developers can be a liberating *Gestalt* switch: the fact that there are now only more and less skilled users leads to a remarkably egalitarian and socially conscious form of design. If developers are users, it is no longer possible to despise the user, or to create software that "hides complexity" in such a way that the user cannot subsequently recover it. Indeed, if users are developers, precisely the opposite is true: it must be assumed that the user's desires are inscrutable, and that she may therefore need to get under the hood *herself* in order to satisfy them.  Likewise, distributing the credit for the software as widely as possible, and taking pains to make users feel like important co-developers by crediting them in the software and including them in the mailing-lists and development discussions, leads to a remarkably cooperative development system.

Taken together these factors form a normative management theory about how to design software – and a very good one at that. This theory, however, does not depend on the existence of the Free Software licenses; it would be entirely possible for this kind of system to operate in a large proprietary software firm, where everyone can see the code and everyone comes to the development meetings—implementing such a management theory on the internet is just a question of scale.[29] Within any given software development firm, the boundaries of the organization would be determined by intellectual property rights—e.g. who can see the source code and who cannot, including such instruments as non-disclosure agreements and limited licensing agreements—and by the information infrastructure—e.g. open or closed mailing lists, access to the code-tree and rights or access to fix bugs or add features. On the internet, all of these issues are organizationally wide open: Free Software licenses allow everyone (anyone covered by contract law) to be a developer, mailing lists are almost always open (though this varies from project to project and depends on the stage of the project—from planning to debugging), and the code-tree is almost always managed in such a way that anyone can submit changes, bug-fixes, ideas. If those ideas are not accepted (say, e.g. Linus rejects your wacky idea for implementing garbage collection in the kernel), then you can take the code and make you own project, and try to attract your own co-developers. There is no management to stop you, and there are no exclusive intellectual property rights to prevent you from doing so.

As Raymond also notes, the other condition that was necessary for this kind of development to occur—alongside the Free Software licenses – was the explosion of access to the internet that occurred around 1993 as a result of the phenomenal growth of commercial Internet Service Providers which extended internet access outside of government and academia, to businesses and individuals: this expansion of the internet made Linux possible—*as a distinctly new species of Free Software*.

The new technical communication situation of the commercial internet—which has its own history and its own set of necessary conditions—produced a radical change in the [ enormously increased ] number of potential contributors to a project like Linux, and as a result, actually increased the eagerness of users to participate in the improvement of software to which they would then have ready, unconstrained access. Users of GNU software (GNU is a recursive acronym: "GNU’s Not UNIX") and other Free Software projects that existed prior to 1993 had been largely academic. The net, such as it was, was neither big nor fast enough to support the collaborative development of a project as complex as an operating system—though this didn't stop the FSF from trying. The FSF was initially founded with the goal of creating an operating system called GNU which would use a "kernel" called "HURD". Hurd has never been successfully completed and what is now commonly referred to as "Linux" began only as a separate kernel, created by Linus Torvalds. Many of the other components of the GNU operating system—such as a compiler, debugger, editor and various tools—were already complete and thence became part of the GNU/Linux Operating System when Linus Torvalds offered the kernel. The resulting explosion of development and integration only took off after 1993.

Raymond's claim in CatB is that the creation of GNU/Linux along with several other, but smaller projects, is an innovation in the process of software development—a better bug-trap. And innovation being what it is to business, such a technique should interest them. Though he doesn't say it, the implication of CatB for the Software industry is that the Open Source Development model is actually a way of using the internet as a system for the efficient allocation of highly skilled labor. The emergence of Linux signals not only a profound challenge to intellectual property, but perhaps more significantly to existing systems of management, hiring, and human resources. If software developers can pick their projects, fix their bugs, and write their features without having to go through management, then the role of management shifts to the person of the project maintainer.

But there is one problem with this understanding of Open Source: almost no one is actually *paid* to write Free Software. It is only in a metaphorical sense that one can currently call the work on Free Software development "employment," since the majority of developer-users are doing it alongside or outside of their official, paying corporate jobs—not as officially employed Free Software programmers[30]

As Open Source catches on in the business world—and it has to a rather phenomenal extent—then the question of the precise mode of remunerating people for their labor becomes a conundrum: why do such highly skilled, employable people create software that they give away without any kind of monetary remuneration? With respect to developers' legal rights, and the actual accountability of software firms to individuals, Open Source could look more like profound exploitation than massive, voluntary, distributed software development.

Raymond is of course aware of this problem and attempts to answer it without any reference to the actually existing world of legal regimes—that is, as an anthropologist for whom informal conventions, and the actual behavior of individuals is more important than the normative sphere of national and international legal structures. Remuneration in this sense, might not simply mean cash-payment, but something less calculated, something he prefers to call a "gift culture" in which informal taboos actually structure the behavior of people more effectively than national or international law.

The following section looks briefly at his explanation. As an anthropological investigation of customary behavior, it is extremely useful; but by strategically ignoring the role of actually existing law—which it should not be forgotten is an essential component of Free Software itself—Raymond misses the opportunity to elaborate the overlap and relationship between the observed customary behavior of hackers and the legal and economic obligation that enfolds them.

On the analogy with anthropology, Raymond sees Hackers like anthropologists once treated the Cuna, or the Trobrianders—as an isolated, functioning societal unit with easily identifiable borders, almost fully disconnected from any legal, economic, or historical realities that structure the contemporary global orders of society. In the case of anthropology, such an assumption has proved impossible to sustain—and it should be even more so in the case of "The Hacker Tribe" which is so firmly and obviously at the center of that legal, technical, and social seat of power. Both the Trobrianders and Hackers exist within overlapping systems of formal legitimate legal systems and informal conventional systems of behavioral regulation.

The task that remains, and that Marcel Mauss initiated under the sign of *The Gift* is to develop an understanding of how the informal taboos and conventions of a given network relate to the formal legal technical structures of property, contract and exchange: it is not a question of communities or societies which have one, and not the other, but a question of how they function together to form an operating system that its inhabitants can and do manipulate.

## Homesteading the Noosphere

> "The 'utility function' Linux hackers are maximizing is not classically economic, but is the intangible of their own ego satisfaction and reputation among other hackers ... Voluntary cultures that work this way are not actually uncommon; one other in which I have long participated is science fiction fandom, which unlike hackerdom has long explicitly recognized 'egoboo' (ego-boosting, or the enhancement of one's reputation among other fans) as the basic drive behind volunteer activity... We may view Linus's method as a way to create an efficient market in 'egoboo' – to connect the selfishness of individual hackers as firmly as possible to difficult ends that can only be achieved by sustained

cooperation." (Raymond, 1997: Section 10).

This quotation from CatB is oft-used in discussions of how voluntary hacker and online cultures function; it suggests that Linus' method—the constructive channeling of the work of many volunteers into a single well-defined goal[31]—is analogous to a spontaneously forming, self-correcting market or ecology.

For Raymond, all possible spontaneous systems are the same: economic markets, ecologies, adaptive systems in biology, the "Delphi" effect—there are references throughout CatB. The minimum characteristics are that all such "self-correcting" evolving systems function the same way: an identifiable and self-directed agent maximizes its utility (or value, or X) through self-interested choices (those choices that lead to a net increase in X); a sufficiently large number of agents doing this in the same system leads not to chaos but to complex, differentiated organization capable of sustainable equilibrium. X in Raymond's case is "egoboo" which he makes synonymous with both "ego satisfaction" and reputation. Raymond does not address what the analogue of equilibrium might be.

The simplification to a quasi-algorithmic description is fine—such mechanisms are entirely common. However, his description of its actual mechanics and context—which might set it off from other market mechanisms or other biological systems—lacks in detail. What should be interesting for such a description are the qualities of and the rules for egoboo maximization; the structure, constraints (either conventional or legal) of the "market" – which requires, as Raymond broadly puts it, “a medium at least as good as the Internet” to function; the necessary qualities of the key figure (Linus, in this case) who we assume must serve some function (he has a "method" after all, so it can't be all madness) in organizing the agents in the market; or even the meaning of stability or equilibrium in this example. For Raymond, there appear to be only two systems in the world, complex adaptive evolutionary "bazaars", and hierarchical, authoritarian corporate "cathedrals". His strategic optimism of course favors the former: proprietary software “cannot win an evolutionary arms race with open-source communities that put orders of magnitude more skilled time into a problem” (Raymond 1997, Section 10).

Raymond specifies some of these qualities in his second paper, “Homesteading the Noosphere” (HtN). It is in HtN that Raymond is at his most anthropological, both in terms of his observations (which are extraordinarily perceptive and valuable) and in his attempts at theorization (which are not). His principal problem concerns the nature of property and space, signaled in the title of the piece: the Noosphere is the "space of all ideas" and hackers are the ones who are homesteading it.

Though Raymond doesn't make it explicit, it is clear that through the metaphor and mantle of anthropology two assumptions are rendered possible:  first is that the borders of the Hacker tribes' space is identifiable and separate from that of contemporary modern society (i.e. the "real world"); the second is that Hackers, as an identifiable and coherent group, have a different, legitimate and perhaps incompatible notion of what property is.  Anthropology is not innocent in rendering possible such assumptions.  Most anthropologists continue to treat indigenous peoples as identifiable and separate based not on "real world" distinctions (e.g. reservations or "homelands" created by present day sovereign nations) but on some combination of kinship, language, culture, and biology.  Likewise the debates over the incompatibility of property regimes (see Brown 1998) strengthen the assumption that such separateness either does or should exist and should therefore be equally legitimate.  In the case of Raymond's anthropology, however, neither of these assumptions hold, but his use of them does in fact reveal very significant details about the behavior of software developers who contribute to software projects.

What I insist is worth paying attention to in Raymond's explanation are the particularities of the relationship between material and immaterial ideas, between writing as a representation of ideas, and writing as a thing in itself—words that do things.  Raymond makes use of four spaces in his article: Noosphere, ergosphere, cyberspace and "the real world". Raymond explains the distinctions thus:

> The 'noosphere'... is the territory of ideas, the space of all thoughts. What we see implied in Hacker ownership customs is a Lockean theory of property rights in one subset of the noosphere, the space of all programs... [Faré Rideau ] asserts that what hackers own is *programming projects*—intensional focus points of material labor (development, service, etc.)... He therefore asserts that the space spanned by hacker projects is *not* the noosphere but a sort of dual of it, the space of noosphere-exploring program projects [ergosphere].
>
> …And the distinction between noosphere and ergosphere is only of *practical* importance if one wishes to assert that ideas cannot be owned, but their instantiations as projects can.
>
> To avoid confusion, however, it is important to note that neither the noosphere nor the ergosphere is the same as the totality of virtual locations in electronic media that is sometimes (to the disgust of most hackers) called 'cyberspace'. Property there is regulated by completely different rules that are closer to those of the material substratum... (Raymond 1998, Section 5)

It is clear from this quotation that the Noosphere, as Raymond understands it, is not like land. Despite his dependence on Locke's philosophy, which was eminently concerned with land (especially certain stretches of good tobacco-growing land in the colonies),[32] Raymond's Noosphere consists of non-excludable, non-material ideas, which can take the particular form of a programming project (an intangible, but perhaps not immaterial thing). Therefore a hacker can *own* this idea in a sense similar to the way a scientist might *own* a research agenda,[33] i.e. it is his only to the extent he can convince others near it not to trespass, either by force or by charm.

What is unclear here is *how* precisely the boundaries of an idea are drawn. Some version of expertise that is shared amongst hackers needs to be already in place: a hacker needs to know how to find, understand, and evaluate what other hackers are working on. There is no equivalent to a fence, a stone wall or a no trespassing sign: rather a hacker is expected, by learned and evolving informal conventional means, to know who owns what.

Of course, the interesting aspect of this proposition is that in the world of hackers and developers, such knowledge of who owns what is relatively robust, even if they cannot articulate exactly *how* they know who owns what. Meanwhile, the actual code that people produce, share, download, archive, compile and run, is in fact *explicitly* (i.e. as part of the code *itself)* identified by a copyright, a name or list of names and occasionally an email or address—and therefore owned in a legal and non-tacit sense. The copyright in the "real-world" represents ownership of the code, even if the idea in informal conventional terms, is understood to be owned by someone else.

Compare this with the division that exists in the "real-world" system of intellectual property. Patents represent ideas, copyrights cover specific materially existing chunks of text. Both must take an explicit written form, though the former is presumed to *represent* the idea, the latter to *instantiate* it: holding a patent means owning an idea, holding the copyright means owning a particular instantiation of the idea—or simply some words on a page. In Raymond's Noosphere the mode of ownership of ideas (i.e. the patent) is *reputation.* Reputation is the proxy for the idea in the same way that the patent specification is the proxy for the idea. But the mode of ownership of the *instantiation* is still copyright: Free software is in fact protected by copyright law, not by reputation or any other non-material patent-like stuff.  This creates an opposition between two spaces: the Noosphere, an imagined communal space where reputation is recognizable but not apprehensible and cyberspace which is where both the written programs and the evidence of reputation (the markers, the discussion, perhaps the *sense* of that reputation) reside.

**Hacker taboos.**

Keeping this set of comparisons in mind, it is illuminating to look at the three basic informal taboos that Raymond has identified in the Hacker Tribe[34].  His point, in HtN, is that these informal taboos are in fact in "contradiction" with the explicit licensing practices.  The contradiction, however, depends on whether or not the realm of informal conventional reputation is seen as part of the same space as formal intellectual property rights.  Since Raymond strategically denies the importance of mere mundane legal rights, and substitutes his speculative "gift culture", these differences appear as contradictions.  The three taboos are:

1) There is strong social pressure against forking a project. It does not happen except under plea of dire necessity, with much public self-justification, and with a renaming.

2) Distributing changes to a project without the cooperation of the moderators is frowned upon, except in special cases like essentially trivial porting fixes.

3) Removing a person's name from a project history, credits or maintainer list is absolutely *not done* without the persons explicit consent. (Raymond 1998, Section 3).

In any given programming project, there is peer pressure against taking the code of the project, and starting a new project. The owner of that original idea accrues reputation in various ways: perhaps through the creation of the initial code-base for the program, through the continued maintenance of the project, or through the management of contributions, changes, or suggestions. Forking a project is discouraged because it dilutes the identity of the project, and could potentially divert reputation from the "owner" of the project. However, forking is precisely what the licenses guarantee *must* be possible in order for software to be copy-lefted.

Compare with the patent. The holder of the patent has absolute rights over its use or reuse. Using a patented idea requires licensing it from the owner. In the Free Software world, however, such a condition has been dispensed with via the hack of copyleft. No owner of a piece of software can prevent you from reusing it—but neither can its reuse be prevented from re-incorporation in the original project.[35]

Forking a software project amounts to the creation of new, equally free, but potentially incompatible versions of the same software.[36] More importantly, it diminishes the brand identity of a single project by giving it *competition*. What Raymond is suggesting with his property analogy is that reputation functions similarly to patent: it grants a limited monopoly, and discourages competition in order to channel reputation—the incentive in Raymond's world—to the owner of the idea. Raymond's free market in ideas is in fact *regulated* by informal conventions, in the same way the real market in intellectual property is regulated by IP law.

The second taboo is essentially the same as the issue of forking, but serves to regulate the behavior of people such that some entity (either a group—the Apache group—or an individual—Linus Torvalds) maintains control over managerial decisions. Authority must emerge somewhere, and it does so through the existence of informal taboos against the anarchic distribution of changes to software. Here the comparison with patent and copyright is apposite and overlapping: the role of patent and copyright is not only to exclude competition from the market for a limited time, but to recognize rights to decide who can or cannot use the intellectual products and for which purposes.

Again, patching software (e.g. the Linux kernel) and releasing the patched version publicly is exactly what the Free Software Licenses are designed to allow. The existence of this convention implies that, for example, the subsequent kernel will not be named "Linux" until Linus or someone else in the hierarchy approves it and incorporates the new code. In fact, interestingly enough, Linus Torvalds holds the trademark to the Linux name—suggesting that even deep within the Noosphere, the regular old real world intellectual property system is functioning to protect the reputation of individuals.

The third taboo is also interesting from a comparative perspective. It suggests that reputation actually depends on its explicit recorded form (what I have called in a separate paper *greputation*[37]*)*. If you are not a project maintainer, but just an aspiring bug-tracker, then your rise in the ranks is dependent on the explicit appearance of your name in the record. In patent and copyright law, the entire range of contributors is rarely given credit (patents more so than copyrights) and the purpose and goal of making these products into property is *to make them alienable*: to provide the ability to erase one name and replace it with another, given an appropriate transfer of some proxy for value (usually: enough money). For Raymond, contributor lists are an informal redistributive mechanism: they portion out some of the reputation that accrues to, say Linus Torvalds, and distribute it to people who have written device drivers, or modules or other less glamorous additions to the Linux kernel.  Again, it results in a "contradiction" because in Free Software licenses, the only name that legally matters is that of the original copyright holder—and this constitutes a market failure, so to speak, that requires a redistributive mechanism which hackers have developed to correct for it.

I should be clear that this comparison with real world intellectual property does not exist in Raymond's explanation—he in fact refers to the very specific legal issues as "hacker ideology" and reduces actually existing license issues to "varieties of hacker ideology (Raymond 1998, Section 2)."  This strategic denial of law and politics is necessary in order to observe the Hacker tribe as it exists in "nature"—the pure realm of the Noosphere where: “Lockean property customs are a means of *maximizing reputation incentives*; of ensuring that peer credit goes where it is due and does not go where it is not due.” (Raymond, 1998).

This formulation, which is clearly intended to have the force of scientific law, is incorrect for a couple of reasons, but nonetheless points to some of the more interesting implications of Raymond's observations.

First,  the property customs  he identifies could more accurately be described as a mechanism to minimize disputes and adequately credit co-developers in a context *outside of any given firm*. Since the bargain of Open Source is that the internet is its medium, and the internet is not a corporate form, dispute resolution needs to take some other form—informal conventions governing idea ownership are perhaps one successful way[38].

Second, these customs do not "maximize reputation incentives." Rather, they are an expression of one optimized design for a structure that would maximize net gain in reputation. One way this is done, outside of deliberate human thought, is through the social enforcement and gradual pragmatic evolution of conventions such as those Raymond identifies.[39]

Furthermore it is not the *incentive* that governs where reputation goes, but rather the *mechanism* of the property conventions themselves. The incentive, such as it is, can only be the *expectation* of what reputation will bring: for example, the power to decide over and maintain a project and to resolve disputes about it. The incentive could also include personal satisfaction, reputation spillover into the "real economy", or simply any subjectively valuable return on the investment of contributing. Everything hangs on what is understood by the term "incentive" here.

One cannot create an "incentive structure" in the sense that economists use that term, without a *measurable* return. And as reputation remains un-measurable, it is not a suitable incentive for such a structure—it remains a metaphor. Indeed, Raymond has identified conventions, which from his extensive experience, actually exist—but there is no evidence that these conventions actually concern reputation, which is an extrapolation on Raymond's part.

However, reputation could be an incentive in a less exact, metaphorical or less material sense: as that return which people *expect* to receive based on their knowledge of the past and their understanding of the structure within which they operate. In this sense, it is part of a structure of reciprocity and obligation whose material substrate is not simply money, but *language*. Or put inversely, the function of money—as a one dimensional measure of trust—can also be served by language. Words do matter then, because they are the medium of reputation, and hence of trust in this system—this community—of individuals who give free software to each other and pay in compliments.

## How to pay with words.

In Raymond's version reputation—unlike money—has differentiated and specific qualities. Whereas money has a single dimension, reputation might indicate any of a range of things: skill, elegance, cleverness, usability, sophistication, training, experience—what is sometimes wittily summarized as "code-Fu". Accordingly, in Raymond's Noosphere, reputation is a better way of both incenting and crediting the authors of code than simply paying them.

But reputation is hard to understand. It is a subtle and incomplete calculus that allows a reputation to form. Raymond likes to insist that good code is obvious because "it works," but this simply passes over the details of how a reputation is formed—much less what it means for code to work.

I would suggest something very mundane here. The way in which reputation is formed—the “allocation mechanism” of reputation—is only *the speech of the participants,* i.e. the things they say to each other to bring each other into line, on line. The lurking romantic author in the world of Hacker software creation may eventually come out to insist that only geniuses—divine beings—write good code. For the rest of us, however, the recognition of reputation is learned, and is a function of trusting what people who other people trust say about themselves and others—it is a hermeneutic and practical experience of reputation.[40] Raymond observes the behavior of hackers, captures the practical essence of this activity, and translate it into rules: don't fork projects, don't distribute rogue patches, don't erase people's names.

In fact, Raymond himself has done more to identify and make explicit these evolving conventions than anyone else; they are nowhere articulated more explicitly than in his published work. It should not be surprising then, that at the end of HtN, Raymond makes a very interesting normative suggestion:

> [Hackers should] develop written codes of good practice for resolving the various disputes that can arise in connection with open-source projects, and a tradition of arbitration in which senior members of the community may be asked to mediate disputes... The analysis of this paper suggests the outlines of what such a code might look like, making explicit that which was previously implicit. No such codes could be imposed from above; they would have to be voluntarily adopted by the founders or owners of individual projects. Nor could they be completely rigid, as the pressures on the culture are likely to change over time. Finally for enforcement of such codes to work, they would have to reflect a broad consensus of the hacker tribe (Raymond, 1998, section 20).

Raymond has effectively proposed what he has already identified as functioning: informal codes adopted by people to manage the direction and control of projects. His identification of the existing codes as "implicit" suggests that people act this way without ever saying anything to each other—but such an assumption is unsupportable. Raymond's specific formulation of them as taboos, in classic anthropological fashion, makes them into regulating rules which the participants themselves rarely recognize (Raymond 1998, Section 2). Raymond suggests that presenting these rules first, as legislative and normative conditions rather than accepting their existence as normative, but informal conventions, will lead to a more robust software development system.

Nonetheless, the suggestion that these rules might govern the development of software projects can only be made under the influence of a fantasy that the Noosphere – the gift culture of hackers – is radically separate from the rest of the “real world.” The only response awaiting such a fantasy is a rude awakening with respect to who exactly the “senior members of this community” are: they are not hackers. The people who get to decide – on anything – are the people who *own* the software; that is, the people who own it in a legal and not in any analogical or metaphorical sense. Right now, the only thing protecting the informal conventions of project management from outside interference is in fact the legal hack of the Free Software licenses. And this hack is effective only so long as the contracts are deemed to be legal and fair—until they are tested in court or in legislatures, and only so long as they are enforced, by whatever means.

Raymond, in fact, knows this, and despite his strategic denial that Open Source software is *not* political, he is also willing to admit that the "right to fork" is like the right to strike or the right to bear arms—rights that constitute the structure within which freedom is possible. Both the Free Software Licenses and the Open Source Definition are intended to ensure the existence of a privatized public domain against the interests of intellectual property-appropriating corporations. This can *only* be political because it concerns the legal constraints on how business is organized and how the US Constitution should be interpreted The effect of using Free Software—regardless of any stated goals—is the political transformation of how business is done and the transformation of the laws which govern commercial activity of software production – *in order to return to software developers the right to make binding decisions about what they create; to take that right away from the patent and copyright owners (e.g. management or shareholders) and give it to the people who* make *and* use *the software —and guarantee them a structural right to maintain this control*. In the end, they are giving each other not software, or value, but *rights.* Rights, in the particular form of contracts, that guarantee *nothing more* than the continued circulation of these rights.

## Part 3: Returning Gifts

There is certainly something puzzling about a bunch of otherwise highly rational software developers making very good software and then giving it away without asking for payment in money. Often, and largely due to Raymond's explanation, the best explanation people can offer is that these people are actually living in a "gift economy".[41] On the one hand, it allows people to recover a moral sense that is assumed destroyed by the money economy: that of trusting fellow members of a community instead of money. On the other hand it is also a way of suggesting that perhaps the economy is evolving into a system where value circulates via different proxy currencies, such as reputation, status, or authorial creativity.[42] In exactly none of these cases has anyone explicitly sought out from the ostensible founder of this notion, Marcel Mauss.

This last section proposes to do just this. However, the goal is not so much to correct Raymond's understanding, or to offer a final scientific explanation of the behavior of hackers that would prove once and for all the Open Source organization is an evolutionarily necessary development. I would rather take what Raymond's explanation and the phenomena of Free Software teach us and *re-read Marcel Mauss*. The benefit of doing this accrues primarily to anthropological science, not to Free Software. That is to say, Free Software clarifies certain aspects of contemporary legal, technical, and political structures in such a way that Mauss' work—and the debates surrounding it—are of renewed importance to anthropological theory.

As we have seen, Raymond's explanation rests on the strategic denial of the importance of the legal sphere. Like most participants in this discussion, Raymond views law as a nuisance—a merely empirical and nearly always misguided application of power to a realm—technology—which is fundamentally unsuited to such regulation. Law and lawyers are parasites on an otherwise evolutionarily necessary technological development, which supposedly obeys laws of a different and more scientific order.

Mauss, on the other hand, quite clearly apprehended the importance of law—especially the regimes of property and contract law—to the constitution of personhood and subjectivity, the behavioral regulation of persons, and the structure of transactions. The view through Mauss sees a world in which the "technical" and the "legal become a much more complex set of structures that govern memory and expectation—structures that organize behavior through techniques of archiving, remembering, anticipating, and expecting.

## How to Read *The Gift*[43]

For Mauss the gift—the empirical moment of a transaction—is a “total social fact.” It is something that leverages all aspects of life together at once. It affects the market, the polis, the home, the university, the court, social structure; it is economic, political, legal, commercial, cultural, social, even aesthetic and religious – all at once. The gift-exchanges of Mauss’ essay, in particular the potlatch,[44] are not irrational economic expenditures, nor are they non-scarcity economic transactions that govern honor and prestige—they govern everything:

> For the potlatch is much more than a juridical phenomenon: it is one that we propose to call ‘total’. It is religious,

> mythological and Shamanist...the potlatch is also an economic phenomenon, and we must gauge the value, the importance, the reasons for, and the effect of these transactions...The potlatch is also a phenomenon of social structure...finally...we must add this: the material purposes of the contracts, the things exchanged in them, also possess a special intrinsic power, which causes them to be given and above all to be reciprocated. (Mauss 1924, p. 38).

This "total system" is a general, observable, organic structure of reciprocity and obligation. Mauss' essay, as Mary Douglas puts it, is like "an injunction to record the entire credit structure of a community." (Douglas, 1990, p. x). This credit structure is one of memory and expectation, both individual and collective; it is embodied in things and manipulated by a combination of actions, names and gestures. It requires for its functioning humans, things (including technical things), and sets of learned rules, conventions and laws of which the gift is an observable index.. The nature of these conventions and laws, their power to coerce and their material substrate all change over time.

Mauss condensed this theory into a single central question: "*[w]hat rule of legality and self-interest, in societies of a backward or archaic type, compels the gift that has been received to be obligatorily* *reciprocated? What power resides in the object given that causes its recipient to pay it back?”* (Mauss, 1990, p. 3).

There are three things which should be noted in this question.

First, Mauss mentions both self-interest and obligation. Self-interest, that which concerns the economizing of a single individual according to personal desires, is present everywhere in his essay but it is always balanced by obligation—in particular by the sanction or censure of *legal* obligation. This fact has been often been missed in discussions about Mauss' essay. Indeed, often the question of altruism—the question of a “pure gift”—is raised instead. This usually leads to research that begins with the question: are the participants in a gift-exchange really “giving” are they simply self-interested calculating individuals, who expect a return on their investments?[45] Jonathon Parry suggests this is a distorting reading of Mauss: “The gift is always an ‘Indian gift’ – that is, one for which an equivalent return is expected – and the notion of a ‘pure gift’ is mere ideological obfuscation which masks the supposed non-ideological verity that nobody does anything for nothing ...this habit of thought has distorted our reading of Mauss’ essay on the gift.” (Parry, 1986, p.455).

The assumption contained in this misreading is that altruism is the opposite of self-interest—but this disregards the fact that an external *obligation* (in the form of legal structures – whether formal or informal) is as obvious, observable, powerful, and explanatory as *revealed* self-interest. Mauss' use of the term "gift" does not imply altruism – in fact he carefully avoids this implication, especially with respect to the potlatch, which he says is "essentially usurious and sumptuary" (Mauss 1990, p 6).

Second, Mauss' central question concerns the status of the *thing* exchanged, and in particular the "power" it possesses to oblige. Throughout Mauss' work, including his studies of magic and sacrifice, his essays on exchange, and his essay on the concept of person, he focuses not on things with special powers(such as fetishes), but rather on the non-obviousness of the distinction between "person" and "thing." When he does ask about this power that the thing possesses, he does not assume that it is a material or magical quality of the thing, but rather, that people treat it as if it has certain powers. Whether they do this as calculating individuals or as ideologically mystified subjects is less important than the fact that this power concerns the empirical organization of things, people, and property/contract laws that form the basis of a social bond—it is the transaction—the gift exchange—which is the observable aspect of this structure.[46]

Third, the phrase "backward or archaic societies" signals Mauss' concern with some kind of evolutionary aspect of the structures of reciprocity and obligation. Despite this obvious concern, his project is not to identify and study "primitive economies" or "primitive economics"; the 20th century discipline of economics is unsuited to capture the "total" nature of the phenomena under study—it must be approached from a more general scientific standpoint. Furthermore, Mauss' interest in the archaic is not an interest in something that is already gone. Rather it concerns "a permanent form of contractual morality. Namely how the law relating things *even today* remains linked to the law relating persons...forms and ideas that, at least in part, have always presided over the act of exchange, and that *even now* partially complement the notion of individual self interest" (Mauss , 1990, p. 4). The role of evolution and change in Mauss is frustratingly unclear. On the one hand, Mauss wants to identify a universal moral element of transactions, a kind of unchanging necessity to human interaction. On the other hand Mauss ends his essay with a set of moral conclusions about contemporary European society that attack the dominant utilitarian ideological position of the day (that of "constant, icy, utilitarian calculation" (Mauss, 1924, p. 76) as one that denies this essential moral necessity. Mauss optimistically forecasts the return of these necessary reciprocal bonds of social solidarity in the early twentieth century rise of mutual societies, social insurance and family assistance.[47]

Nevertheless, Mauss' moral conclusions are based on his assertions that there is an evolution to the relationship between persons, things, and rights: namely the displacement of an ancient, or archaic system of gift-exchange  by a modern one based on individualized *contract* (beginning with Roman law and the legal distinction between persons and things) and especially, on *money-exchange.*

After posing his central question, and raising the awkward issue of an quasi-evolutionary development, Mauss repeatedly stresses his method of answering it. Mauss' essay is famous for the scholarly apparatus that dominates it. For Mauss, the "credit structure" of a society is to be uncovered by a "method of exact comparison." (Mauss, 1990, p. 4). It is to be found "through documents and philological studies of the consciousness of the societies themselves, for here we are dealing in words and ideas" (Mauss 1990, p.4-5).   Etymology (wherein the specific relational meanings of terms can be discovered), comparative history of law (especially concerning that of property and contract – and especially in societies where written law is an institution), and ethnographic report (in societies where writing was less easily accessible, or perhaps unavailable) fill the notes that range around the world and deep into the past.

The breadth and depth of Mauss' knowledge are often noted, but rarely is it recognized as a particular species of comparative research. Mauss was engaged in creating a version of historical and anthropological research that investigates the structures of reciprocity and obligation (esp. of property and contract) across societies—from the tribes of Melanesia and the Northwest Coast to the societies that form the basis of European civilization: Hellenistic, Roman, Indo-European, Germanic and Celtic. These “archaic” societies, and the “exact comparison” that Mauss seeks to perform on them, constitute not the “other” to, but the foundation of the present.

Mauss explains his method this way: “we shall arrive at conclusions of a somewhat archaeological kind concerning the nature of human transaction in societies around us, or that have immediately preceded our own.” (Mauss, 1924, p. 4). Levi-Strauss recognized in Mauss' method a profound shift in social scientific observation—away from a mere cataloguing of forms towards a more scientific elaboration of social structures. Levi-Strauss' "rescue" of Mauss' method resulted in the creation of structuralism (Levi Strauss 1987, 45-66). Ironically, Mauss' method might also be seen as the origin of what Foucault would later describe—in opposition to structuralism—as his archaeological method of historical research.[48]

## Money, gifts, and laws as technical things

What Mauss seeks to describe in *The Gift* is a *technical* structure of reciprocity and obligation. The word "technical" can be misleading here, and perhaps should be specified in the following way.[49] The structure of obligation and reciprocity is the total system of memory and expectation within which people, things and rights circulate—it includes speech, gesture, and written communication, as much as it does human organisms or comestible things. Memory – both biographical and technical – concerns archiving, remembering, recalling, keeping accounts, eulogizing, writing history, etc.; expectation – both biographical and technical – concerns anticipating, planning, hoping, expecting, and deciding. In *The Gift*, this changing technical system has at least two very important observable, material aspects: gift-exchange and money-exchange.

"Money," in *The Gift* is not price, Mauss is not writing about the introduction or evolution of a price system. For both the *material* aspect of money and its essential *relationship* to a (legal) regime of individualized contracts are important. In the purest version of price theory, money is has neither quality nor quantity because it has no effect on the pure ratio between prices. However, financial economics and institutional economics both stress the importance of money as a part of the economy, and return it to the model as an essential empirical given. Martin Shubik puts it concisely: "Much of micro-economic theory has been devoted to the mysteries of maximizing high-dimensional utility functions… Perhaps a more natural view of a modern economy is that money is the reality, and the utility function is the shadow abstraction."(Shubik, 2000, p.19).

"Money" is also not a "commodity"—which is a confusion often made by anthropologists (see Gregory, 1982; Gregory, 1980; Carrier, 1991)—especially not "commodity" in the sense given by Marx – about whom Mauss, curiously, has almost nothing at all to say. A comparison between Marx's use of "fetish," its role in his definition of commodities and Mauss' descriptions of money (for which he also uses various foreign words for magic—such as *mana*) should be done—but it is outside the scope of this paper. Nonetheless, there is no hard distinction anywhere in Mauss between "gifts" and "commodities," only between money and gifts. Simply replacing money, then, with value or commodity or price obscures rather than enlightens.

To put a finer point on what Mauss' means by money, there are two sections worth quoting here:

> "In these societies we shall see the market as it existed before the institution of traders and their main invention – money proper. We shall see how it functioned both before the discovery of forms of contract and sale that may be said to be modern (Semitic, Hellenic, Hellenistic, and Roman) and also before money, minted and inscribed." (Mauss, 1990, p. 4 ).

And later:

“These phenomena [potlatch; gift; circulation of gifts, persons, and rights] allow us to think that this principle of the exchange-gift must have been that of societies that have gone beyond this system of 'total services' (from clan to clan, and from family to family) but have not yet reached that of purely individual contract of the market where money circulates, of sale proper, and above all of the notion of price reckoned in coinage weighed and stamped with its value.” (Mauss, 1990, p. 46).

Two things are clear in these passages

First, money is a *medium*, a material thing that is weighed, stamped, inscribed, and related to explicit and implicit contracts. It is at the same time a thing and sign. It is not alienated, inasmuch as Mauss withholds judgment on who, or what, it would be alienated from. Mauss sees money as precisely one kind of contingent *institutional* invention in a historical frame, not as an inevitable evolutionary development of trade. Such a focus on the technical nature of the institution is not arbitrary, but constitutes the particular focus of a very long footnote (a footnote with a *title* no less) on the subject of what makes money money.[50] Mauss explains here that the inscription on most money (e.g. “our” money—meaning European money) is the symbol of the State, of a particular sovereign guarantee. When

Malinowksi suggests that the Melanesians do not have money, it is this particular form that Mauss accuses him of elevating to a general type. On the contrary, explains Mauss, the Melanesians, even in Malinowski's study, possess things that *function exactly like* money: "They have purchasing power, and this power *has a figure set on it*... the idea of number is present even if that number is fixed in a way different from that of the state, and varies during the succession of kula and potlatches." (Mauss, 1990, p. 101).

For Mauss, money is both a thing and a sign, marked with a number. Its use, function, inscription and characteristics indicate only the specific institutions that guarantee or fix this thing/sign (i.e. in the case of "our" money, the State, or today—currency markets; in the case of the Melanesians, the relative positions and ranks of gift-givers). Thus Mauss' study concerns the development of money as the medium which structures and is structured by the legal regime of property and contract, i.e. the circulation of people, things, and rights.

And while it might be said that gift-exchange is an expression of one configuration of property rights and contract systems and that money is another, or a subsequent one, it is clear, even to Mauss, that such a distinction will not hold. Money starts to infect gifts even at the supposedly archaic stage before money is invented. Certainly, gift-exchange and money-exchange have *coexisted*, but the study of their relative histories is not elaborated by Mauss.[51]

Second, these quotations suggest that there is a progression from exchange-gifts as total-system to the system of money and individual contract. Mauss sees the institution of money and the individualization of sale contract as representing a certain triumph over the previous system of obligation and reciprocity. Though it remains undeveloped in his work, the implication is that the money-exchange system is no longer a total system of services, as the archaic gift-exchange system was.  Today, many things supposedly happen outside of the exchange of money—that is outside the market proper: for example mate-selection and marriage, or voting, or the array of things *that we now refer to as gifts.* The notion of "gift," as we understand it today, i.e. as something that you receive but don't pay for (or perhaps: something you don't deserve), can be seen as an *effect* of the disappearance of gift-exchange as a total-system of services.

If such an evolution has actually occurred, then this is also where the notion of altruism and the debate about a "pure gift" would emerge alongside a now nearly undeniable suspicion that exchange is always interested.  Here, if anywhere, is where the "free gift" – the AOL CD, the free mobile phone, the sample medicine, the free browser – shows its true color as an object with interests attached—as a deliberate manipulation of non-market exchange in the service of market exchange.  It is Mauss' suspicion that the supposedly eternal truth that “there is no such thing as a free lunch” actually has an evolution to it, and that this can be uncovered in the technical structure of memory and expectation; in particular in the history of property and contract laws which  have individualized to such an extent that the nature of trust itself has been transformed.

## Memory and Expectation or, How to do things with Mauss

If we return to the present context of the internet and Free Software, then we can make some broad and initial suggestions of what aspects of contemporary life can be understood as aspects of the technical structure of memory and expectation.  Memory as I have used it throughout this article, refers to the whole structure of memory –  not the biology of the human brain. The technologies of archiving, accessing, recalling, honoring both people and events, the states of mourning, eulogizing, and commemorating; the creation of tradition out of the iterative recognition of an event; and perhaps most importantly today, the writing of history.  This structure could be seen to include everything from libraries and postal systems to pen-pals and Stasi files.  It is perhaps "the public sphere" if this abstraction didn't reduce communication to ephemeral speech.

Consider a quotation from Derrida's *Archive Fever* (Derrida 1995). Here, Derrida imagines what Freud's psychoanalytic theory would have been like if Freud had access to "MCI and AT&T phone cards, portable tape recorders, computers, printers, faxes, televisions, teleconferences, and above all E-mail." Not only do these technologies allow for faster, better, more complete, more accurate recording, they alter the structure of the archivable content and in turn the structure of our experience of it. "the technical structure of the *archiving* archive also determines the structure of the *archivable* content even in its very coming into existence and in its relationship to the future." (Derrida 1995, 16). Furthermore, such a technical structure is not a Kantian category, a master market, or a phylogenetic past, but it concerns something more mundane and approachable:

> The example of Email is privileged in my opinion for a more important and obvious reason: because electronic mail today, even more than the fax, is on the way to transforming the entire public and private space of humanity, and first of all the limit between the private or the secret (private or public), and the public or the phenomenal. It is not only a technology, in the ordinary and limited sense of the term: at an unprecedented rhythm, in a quasi-instantaneous fashion, this instrumental possibility of production, of printing, of

conversation, and of destruction of the archive must inevitably be accompanied *by juridical and thus political transformations. These affect nothing less than property rights, publishing and reproduction rights*... To put it more trivially: what is no longer archived in the same way is no longer lived in the same way." (Derrida 1995, p.17-18, emphasis added).

The archive Derrida refers to here is part of the total archived memory of human societies, especially to the now very significant extent that the internet reaches, or is felt, almost everywhere. This memory, is not just the conservation of the past, as if in bedrock, that we hope will always be there for us to recover. Rather it also concerns the *production* of that memory—the archaeology, the science, and the writing that organize it as *events* which we must also live with—which in turn structures *expectation.* This active archive that produces events (i.e. structures reciprocity and obligation) was what Mauss sought to investigate in the history, etymology, and philology of law and exchange. It should be no surprise then, that Mauss should focus on money, in particular: it is perhaps the most powerful simplified and generalized technology of memory and expectation[52]. Money reduces memory and expectation to a single number: how much money I have now, and how much money I expect to have in the future. Its legitimacy reduces trust to a tangible, material, moveable thing. Thus can I make decisions. It removes authority from individuals and puts it on paper. It thus demands a philological approach.

One can also ask a version of this counterfactual ("would psychoanalysis be different if Freud had used email?") in contemporary terms: how does the renewed contemporary importance of *reputation* relate to the technologies that now organize our lives, and how is it different from money? The world-wide, always-on, constantly shifting network of servers, hard-drives of archives (the archive drive?), communications both controlled and monitored by software (some Free, some not) affects the living organisms that depend on the circulation of just this information in order to know each other and make decisions about each other. The experience of being reputable has changed as the technologies for measuring and producing reputation have also changed. Rating systems, trust networks, and collaborative filtering have become ubiquitous features of our use of the internet, and there is no way to keep that separate from our perceptions and experiences of the non-internet—if such a place still exists.[53]

Perhaps this counterfactual is a useful way of querying Mauss and Free Software together. For Mauss, software would be an unknown, impossible factor in a world of gifts and money governed by the gestures and demands of humans. Things are no more or less obvious today—and the bewildering fact of Free Software and its manipulation of the legal system is equally worth confronting with the seemingly unrelated insights of Marcel Mauss.

**What you get when you give Free Software?**

Rights.

Transacting, whether via money, reputation, or any other tangible or non-tangible proxy, is the circulation of rights. This fact, which Mauss tried to explain via the notion of *hau* is actually made tangible and specific in the case of Free Software.

The point has often been made that the Open Source development model does not *create* software, it only perfects it. And no matter how much reptation is on offer, creating software still requires some sustained, concerted effort on the part of individuals, universities, organizations, or corporations. Someone must make something that demands to be made (a developer must scratch an itch, an entrepreneur must identify a need) in order to give it away. There must be some tangible, believable mode of *expecting* that this effort will not be in vain. Or, put differently, there must be an expectation that such a gift will be reciprocated, directly or indirectly: through monetary remuneration, through reputation remuneration, or perhaps through the sustained and self-reproducing effort of further software production. There must be, in some generalized sense, a return on investment. However, no one is required or expected or asked to pay *anything* (money, reputation, software) for this software or this effort.

Alternately, the gift of Free Software can be experienced as just such a reciprocation: people create Free Software because they *owe* something; many hackers often say something like “I just want to *give something back*"—and often they wish to give back to the *internet in general, not to any particular person.* The debt of having been given so much good high quality stuff without any strings attached, eventually strikes some people as something they have not yet paid for. And yet paying, by the very nature of the definition of Free Software, is not required in any legal or technical way. Individuals are free to take as much software as they want, without end. As long as they keep giving it away, they *never* have to pay for it.

In Mauss' terms, there is some "permanent form of contractual morality", that governs this tension. In short, there must be something that connects the making of Free Software to its using: its *hau*.

Perhaps the most commonly cited, reviewed, investigated, critiqued and quoted section of *The Gift* is not by Mauss at all. In fact it is not even a citation of one of Mauss’ informants. Rather, it is the section attributed to the Maori informant of Elsdon Best, Tamati Ranaipiri. It is a contentious passage: Levi-Strauss chastises Mauss for being duped by Ranaipiri's explanation of the *hau*, Marshall Sahlins insists on retranslating the passage and interpreting "hau" as "profit" and Derrida attacks Levi-Strauss for misreading the passage in a "linguisticist manner" in order to complete structuralism.[54]

Mauss' own reading of this passage (Mauss 1990, p. 11-12) focuses on several important words: *mana, oloa, tonga, taonga* and *hau*. Among these, Ranaipiri has used *taonga* and *hau*. *Hau*, Mauss suggests, is like the Latin *spiritus* (perhaps, the sense of German *Geist*, or French *esprit* better captures it, than English *spirit*), meaning both wind and soul (Mauss, 1990, p. 89 n.26), while Ranaipiri has said "the *hau* is not the wind that blows – not at all."  *Taonga*, as Ranaipiri uses it, suggests any old thing; an article that can be transferred, but Mauss is at pains to point out that *tonga*, or *taonga* are the sacred objects of families or clans, immovable property that is separated from the family, the clan or the tribe only under severe conditions; it is most certainly not the everyday *oloa* of barter (Mauss, 1990, p. 9-10). Indeed, in a footnote he stresses: "the *taonga* seem to be endowed with an individuality, even beyond the *hau* that is conferred on them through their relationship with their owner. They bear names." (Mauss. 1990, p. 91 n. 32)". The *taonga* are specific kinds of goods, animated by this thing called a *hau*, which Ranaipiri explains as follows: "The *taonga* that I received for these *taonga* must be returned to you. It would not be fair (*tika*) on my part to keep these *taonga* for myself. I must give them to you because they are a *hau* of the *taonga* you gave me."

Mauss, almost as if he were mocking his own eagerness to see something "of capital importance" in this text, says that Ranaipiri's explanation "*gives us*, completely by chance, and entirely without prejudice, the key to the problem." (Mauss, 1990, p. 11, emphasis added). I cite it here in pieces, in order to draw special attention to how Free Software follows this logic almost exactly. I want to offer the provocation that the *hau* which binds the giving (making) and taking (using) of Free Software together is the *license itself*, not the software; this "technically" unimportant comment code, which is "legally binding" –individually, socially and legally obligatory.

The licenses are the *taonga*, the sacred property of a clan, for an era when the clan is the size of the internet and the world it covers. They are rights, which circulate, and in order to remain rights and not become mere *oloa*, they must continue to circulate, and obligate themselves to do so in their very text.

> What imposes obligation in the present received and exchanged, is the fact that the thing received is not inactive. Even when it has been abandoned by the giver, it still possesses something of him... (Mauss 1990, p. 12)

The software must contain a copyright and a name—without it there is no possibility of contractual obligation—only abandoned text. This copyright and name follow the software wherever it goes.

> ...It not only follows after the first recipient, and even, if the occasion arises, a third person, but after any individual to whom the *taonga* is merely passed on... (Mauss 1990, p. 12)

Anyone who distributes it *must* sign the license in order to pass the software on. It is not possible, legally speaking, to distribute the software without signing the contract. Furthermore, the license itself, which is *taonga*—sacred property—cannot be changed, and requires itself to remain with the software.

> ...the *taonga* or its *hau* – which moreover possesses a kind of individuality – is attached to this chain of users until these give back from their own property, their *taonga*, their goods, or from their labor or trading...this in turn will give the donors authority over the first donor, who has become the last recipient (Mauss 1990, p. 12).

The "users" (Mauss' word), are obligated to give the same rights to the next user. No one is privileged in this exchange. Modified software originally owned by one person must be distributed *even by the owner* according to the original license –

and all other software created in such a manner works the same way.[55]

If Free software licenses are the *hau* that compels the recipient to return something, and thus sediments its status as a particular kind of sacred property, then they are not alienable from the community that possesses them, i.e. the people who have decided to submit themselves to the use and development of Free software *for whatever reason*.

The legal hack of the Free Software Licenses does not explain all of the reasons why people make things, buy things, steal things, or give them away. Informal economies of openness and sharing exist everywhere and continue to be created; and this is a very good thing. Communities of people who possess partial shared commitments to only vaguely understood goals constantly find each other and share resources with each other without ever having to think about the intellectual property or contract law systems. And so it should be.

But Free software is a way of trying to make these communities *scale.* That is, to allow any potential person or thing anywhere to join such a community of people and things and participate in it – to *expect*, to *anticipate* that they will not be excluded because they have not paid for the resources or been hired (accepted) by those who possess the resources. The licenses are part of the thing itself, if software is a thing. They animate the thing and obligate the person. Their existence and their success allows people to put some trust in them, to act as if they will always have this power and to contribute to their further circulation. People write free software, then, because they recognize some-thing of inalienable value to the community in an age when the community is the whole world.

## ACKNOWLEDGEMENTS

An earlier version of this paper presented to the American Anthropology Association in November 2000. Thanks to Thomas Hadsell Henry-Chivens, David Hess, George Marcus, Matthew Price, Michelle Murphy and Hannah Landecker for helpful suggestions and revisions. This paper was originally solicited, edited, vetted and accepted for publication by the JAI Press quarterly series *Knowledge and Society*, but was ommitted from the final publication. Some of the editorial work on this paper was perpetrated by the editors of this journal.

## LICENSE

## REFERENCES

Appadurai, A. ed. 1986. The social life of things: Commodities in cultural perspective. Cambridge University Press.

Austin, J.L. 1962. How to do things with words. Harvard University Press.

Barbrook, R. 1998. The High Tech Gift Economy. Firstmonday December 1998. (http://www.firstmonday.dk/issues/issue3_12/)

Becher, T. 1989. Academic tribes and territories: Intellectual enquiry and the cultures of disciplines. Taylor & Francis.

Boyle, J. , 1996. Shamans, software and spleens. Harvard University Press.

Brown, M. F. 1998. Can culture be copyrighted? Current Anthropology, 39(2), 193-222.

Carson, A. 1999. Economies of the unlost. Princeton University Press.

Coleman, E.G. 2001. The politics of survival and prestige: Hacker identity and the global production of an operating system. Manuscript, on file with author.

Dibona, C., S. Ockman, and M. Stone, 1999. Open sources: Voices from the open source revolution. O'Reilly & Associates.

Commons, J. R. 1968. Legal foundations of capitalism. University of Wisconsin Press. (Original date 1923).

Coombe, R. 1998. The cultural life of intellectual properties: Authorship, appropriation and the law. Duke University Press.

DeLaet, M. and A. Mol, 2001 ??

Derrida, J. 1995. Archive Fever: A Freudian impression. Chicago University Press. Trans. Eric Prenowitz.

Derrida, J. 1992 Given time I. Counterfeit money. University of Chicago Press. Trans. Peggy Kamuf

Douglas, M. 1990. “Introduction.” In M. Mauss The Gift: the form and reason for exchange in archaic societies (W. Halls, ). W.W. Norton.

Evers, S. 2000. An introduction to open source software development. Diplomarbeit, Technische Universitaet, Berlin Germany. On file with author.

Foucault, M. (1972) *The Archeology of Knowledge.* New York: Pantheon.

Ghosh, R. Prakash, V. V. 2000. The orbiten free software survey. firstmoday. (http://www.firstmoday.dk/issues/issues5_7/ghosh/index.html)

Ghosh, R. A. 1998. Cooking pot markets: an economic model for the trade in free goods and services on the internet. firstmonday. (http://www.firstmonday.dk/issues/issue5_7/index.html)

Grassmuck, V. 2000. Freie software: Geschichte, dynamiken und gesellschaftliche bezüge. (Available at http://mikro.org/Events/OS/text/freie-sw.pdf, and on file with author)

Gregory, C. 1980. Gifts to men and gifts to god: Gift exchange and capital accumulation in contemporary papua. Man, 15(4), 626-652.

Gregory, C. 1982. Gifts and commodities. Academic Press.

Hagstrom, W. O. 1982. Gift giving as an organizational principle in science, in B. Barnes and D. Edge ed. 1982. Science in Context: Readings in the sociology of science. MIT Press.

Horowitz, M. 1992. The transformation of American law 1870-1960. Oxford University Press.

Kelty, C. 2000. Scale and convention: Programmed languages in a regulated america, MIT.

Klein, N. 2000. No logo: taking aim at the brand bullies. Picador USA.

Kuwabara, K. 2000. Linux: A bazaar at the edge of chaos. firstmoday. (http://www.firstmoday.org/issues/issues5_3/kuwabara/index.html)

Lerner, J. Tirole, J. 2000, march. The simple economics of open source. (NBER Working Paper No. W7600 http://papers.nber.org/papers/w7600)

Lessig, L. 2000. Code and other laws of cyberspace. Basic Books.

Levi-Strauss, C. 1987. Introduction to the work of marcel mauss (F. Baker, ). Routledge and Kegan Paul.

Levi-Strauss. C. 1992. Tristes Tropiques. Penguin Books. Trans. John and Doreen Weightman. (Original year, 1955).

Levy, S. 1984. Hackers: heroes of the computer revolution. Anchor Press/Doubleday.

Lewis, D. K. 1969. Convention: a philosophical study. Harvard University Press.

Malamud, C. 1992. Exploring the internet: A technical travelogue. PTR Prentice Hall.

Mauss, M. 1990. The gift: the form and reason for exchange in archaic societies (W. Halls, ). W.W. Norton.

Mauss, M. 1950. Sociologie et anthropologie. Press Universitaires de France

Mauss, M. 1974. Oeuvres Vol. 2. Les Editions de Minuit.

Merton, R. K. 1973. The sociology of science: Theoretical and empirical investigations. University of Chicago Press.

Moody, G. 2001. Rebel code. Linux and the open source revolution. Perseus Press.

Parry, J. 1986. *The Gift,* the indian gift and the `indian gift'. Man, 21(3), 453-473.

Polanyi, K. 1957. The Great Transformation. Beacon Press.

Raymond, E. 1996. The new hackers dictionary (3 ed.). MIT Press.

Raymond, E. 1997. The cathedral and the bazaar. (see http://www.tuxedo.org/~esr)

Raymond, E. 1998. Homesteading the noosphere. (http://www.tuxedo.org/~esr)

Raymond, E. 1999a. The cathedral and the bazaar. O'Reilly & Associates.

Raymond, E. 1999b. The magic cauldron. (http://www.tuxedo.org/~esr)

Rheingold, H. 1993. The virtual community: homesteading on the electronic frontier. Addison Wesley.

Sahlins, M. 1972. Stone age economics. Aldine Press.

Shubik, M. 2000. The theory of money and financial institutions. MIT Press. (Vol. 1)

Sklar, M. 1988. The corporate reconstruction of American capitalism, 1890-1916. Cambridge University Press.

Stephenson, N. 1996. Mother earth, motherboard. Wired, 4(12). (http://www.wired.com/4.12/motherearth/)

Stephenson, N. 1999a. In the beginning was the command line. (http://www.cryptonomicon.com/beginning.html)

Stephenson, N. 1999b. Cryptonomicon. Arrow Books.

Tuomi, I. 2000. Internet, innovation, and open source: Actors in the network. First Monday. (http://www.firstmoday.org/)

Wayner, P. 2000. Free for all: How Linux and the free software movement undercut the high-tech titans. Harper Business Press.

Weinberg, G. 1971. The psychology of computer programming. Van Nostrand Reinhold.

1 Free Software and Open Source are two words for the same thing, as I explain below. I tend to use the former, except when referring to the Open Source organization.
2 There are several levels to when code is source, or object, and object code can always be source code for more object code. One particularly illuminating example of this is the legal debate over the code for the DeCSS decryption algorithm for DVDs, which Carnegie Mellon Computer Science professor David Touretzky has subjected to a variety of reformulations, in order to highlight the tenuousness of the distinction and its relation to 'free speech'. See http://www.cs.cmu.edu/~dst/DeCSS/Gallery (visited April 15, 2001) for more details.
3 One could conceivably patent the code rather than copyright it, but the process is prohibitively more difficult and expensive. Copyright is free and easy and can be asserted without going through a bureaucratic process.
4 In theory, it shouldn't matter, and in fact, since the ostensible purpose of the patent system is to make inventions available to the public domain, after the term of protection is up, the source code would be a more accurate and useful representation of most software inventions. However, it is possible—as independent innovators are occasionally encouraged by patent strategy handbooks and IP lawyers—to submit the object code (i.e. the compiled code written in hexadecimal notation on microfiche—which must therefore be decompiled to be read or understood) to the Patent Office precisely in order to obfuscate the actual implementation of the idea; that is, the ideology of patents applied to software is that it must be made publicly available as an invention, but marketplace strategy dictates that inventors make it very, very difficult for someone to actually use the invention, to prevent legal, sufficiently different implementations from bypassing the patent. I am grateful to Wynship Hillier for his information and experience on this topic.
5 See (Brown 1998) for an interesting discussion of intellectual property and indigenous cultural products.
6 See Naomi Klein's *No Logo* (Klein 2000) for the evolution of trademark and brand strategy in the corporate world.
7 Recent work in Science and Technology Studies (deLaet & Mol 2000) focuses on this issue. This work queries definitions of 'technical', 'working' and the boundaries of a technology (manuals, support and as in this volume, the meaning of technology transfer[??]).
8 Since software usually consists of hundreds of separate files, the actual license is generally included as a separate document called COPYING or README in the top-level directory of the software package.
9 Copyleft is a pun the Free Software Foundation uses to refer specifically to the General Public License and similar licenses. Software is only copylefted if the contract specifies that all subsequent derivations must also be copylefted (the FSF is very fond of recursion); thus other licenses such as the BSD license are not copyleft, because modified versions do not need to be re-released under the BSD license (See n.11 below)

10 In the US, contract laws are somewhat less powerful since they do not derive directly from the constitution, as intellectual property law does. In the US, The Uniform Commercial Code (UCC) governs all of the aspects of any commercial transaction, from buying a candy bar to signing a multi-year production contract.
11 There are some important but subtle differences amongst the various available licenses that qualify as Free Software. In particular "BSD-Style" licenses do not have the same provisions, and thus, modified versions do not have to maintain the same contract. This is closer to Public Domain software, but allows the copyright holder to maintain minimal control over it. Still other licenses have different terms. See http://www.fsf.org/ for more details on the specifics
12 There is a double entendre here: hacking into the law, as well as hacking legally. The fact that the word "hacker" lives a public life as meaning "breaking into or criminally defacing private property" is generally a misunderstanding of the mainstream media. Hackers themselves often insist on differentiating themselves from "crackers" who are the supossedly malicious, adolescent law breakers. This subtle differentiation is worth study of its own, but is beyond the scope of this paper.
13 Indeed, many people confuse Free software with Freeware or Shareware, programs that are distributed either for free or for a small fee, but for which the source code is not available, and is not governed by a license like the GPL.
14 Confusion also exists between “open standards” and “open source software.” The standards and the protocols that make the internet what it is are not Free software and are most often not copyrighted. They are in the Public Domain and freely downloadable from the internet. The difference is the following: it is possible for Microsoft to, as they put it, “embrace and extend” certain web and internet protocols. They can take a public domain protocol, incorporate its requirements and build their software to its specifications, then add other “standards” – which they call “features” – to subsequently make it strategically incompatible with systems that do not use their “complete standard.” As a result, people must purchase Microsoft products in order to collaborate with other people who use Microsoft products, while the original standard languishes in the “Public Domain.” Ironically, then, standardization can cause incompatibility. Free software programs can most certainly do the same thing, except that since the source code is open, users could conceivably modify it to conform once again to the standard. It encourages competition in the standards domain, as opposed to oligopolistic (or monopolistic in this case) control of standards through protection by intellectual property law. Open source, as will become clear later in the paper, is another name for Free Software, and does not necessarily rely on open standards.
15 Anyone is crucial here. It does mean anyone – corporation, government, individual, dog. It also means that each particular anyone has access anonymously—and this is a technical term which implies that one does not need to trade personal information in order to download something.
16 Rheingold's book is available online at http://www.rheingold.com/vc/book/ (visited 15 April 2001) (Rheingold 1993).
17 A term Geert Lovink has used to describe the madness of 'Dotcommania'.

18 His web page suggests the following plugs:
Eric S. Raymond is an Internet developer and writer living in Malvern, PA.
Eric S. Raymond is a wandering anthropologist and troublemaking philosopher who happened to be in the right place at the right time, and has been wondering whether he should regret it ever since.
Eric S. Raymond is an observer-participant anthropologist in the Internet hacker culture. His research has helped explain the decentralized open-source model of software development that has proven so effective in the evolution of the Internet. His own software projects include one of the Internet's most widely-used email transport programs. Mr. Raymond is also a science fiction fan, a musician, and a martial artist with a Black Belt in Tae Kwon Do. His home page is at URL:http://www.tuxedo.org/~esr.
19 There are, of course, a number of anthropologists studying "cyberculture" loosely defined, but only a handful of academics who have written specifically about Free Software or Open Source (Evers 2000, Grassmuck 2000, Coleman 2000, Tirole and Lerner 2000, Tuomi 2000, Kuwabara 2000). A recent book by philosopher Pekka Himanen addresses similar issues (Himanen 2001).
20 The original Jargon file is maintained and updated in several places online (including http://www.tuxedo.org/jargon). For the offline version see Raymond, 1996. The original "Hackers Dictionary" was published by Guy Steele, another longtime maintainer of the Jargon file.
21 See e.g. Levy 1984, Dibona 1999, Moody 2000, and Wayner 2000.
22 Two years later, the US Justice department split up Microsoft, by ruling that their strategic confusion of "browser" and "operating system" was tantamount to monopolistic manipulation of the browser market.
23 Indeed, the change-log for CatB proudly displays the replacement of "Free Software" with "Open Source," see the online version (Raymond 1997).
24 see Stephenson 1999a for the Geek court poet's ode to the "command line" and Linux.
25 It was at this moment that business plans and the question of how one can make money on Free Software became an obsession with the media. Prior to this, one of the rare business-oriented Free software companies was Cygnus, now part of Red Hat, whose clever slogan was "Making Free Software more affordable" See Dibona 1999 for more on this history.
26 A certification mark is a species of trademark in the US Patent and Trademark system. Because "open source" is a descriptive term covering a type of product and not the brand name of a specific product, the Open Source organization was forced to go with the certification mark, though they originally envisioned using a trademark originally. See http://opensource.org/docs/certification_mark.html (visited April 16, 2001), for information. Compare this also, with the ownership of trademark by individual project managers on the name of the software (such as Linus Torvalds, who owns the trademark for Linux)—this is discussed in more detail below in section XX.

27 The list of licenses approved by both organizations continues to grow, and as of this writing contains only a few licenses that are contested. See http://www.fsf.org/philosophy/license-list.html (visited 19 January 2001) and http://www.opensource.org/licenses/ (visited 16 April 2001) for comparison. The differences largely concern issues of compatibility – the combination of software licenses under subtly different terms could produce rather hairy issues of legal wrangling, issues that the original GPL, with its strong requirements concerning re-use and modification, was designed to avoid.  However, as none of these licenses has been officially tested in court, such issues remain unresolved.

28 “Experimental” is the wrong word. Though in suggesting scientificity it suits Raymond’s purposes. The “experiment” was actually his “experience” as the leader of the open-source fetchmail project; he called it a “deliberate test of some surprising theories about software engineering suggested by the history of Linux.” This is experimentation only in the not unrelated sense that one says “Eric is experimenting with LSD.” His 'experience' is nonetheless enlightening.

29 Raymond cites Gerald Weinberg’s classic *The Psychology of Programming* (Weinberg, 1971) for its recognition of the evils of “territoriality” over code within software firms; the suggestion being that software development methodology within firms has previously recognized the problems which open source internet-based software development now solves.

30 Though anecdotal evidence suggests that this number is rising quickly as corporations recognize either the value of having such expertise on staff, or in some cases, feel a duty to remunerate in general for the creation of this software—this latter is addressed explicitly in Raymond's third paper "The Magic Cauldron" (Raymond 1999b).

31 Raymond reproduces Gerald Weinberg's citation (Raymond 1997, Section 10) of Kropotkin on the difference between "ordering, scolding, punishing, and the like" and the "severe effort of many converging wills".  Together the three of them manage to produce a believable fiction that there is such a difference at work in the management of software development projects.  The revolutionary analogy, however, is simplified to a scientific principle as quickly as Kropotkin elevated it to a moral one  (See Weinberg 1971).  Much remains to be said on the role of Anarchist thought in the technolibertarian stream of Hacker cultures.

32 Raymond's use of Locke's "Common Law" doctrine is intended to validate something about the Noosphere.   Compare Barbara Arneil's version of the historical roots of Locke's theory, in which his involvement with the Earl of Shaftsbury's colonial endeavors played an important role.  See (Arneil 1996)

33 Science and technology studies and the sociology of scientific knowledge have long studied such problems.  One example could be Robert K. Merton's use of "intellectual property" to refer to a scientists reputation for a particular result or experiment (Merton 1973).  More specific—and apropos given the anthropological metaphor—is Tony Becher's *Academic Tribes and Territories* in which he discusses the role of the "people-to-problem ratio" by using the metaphor of urban and rural populations—some problems are over-studied, crowded and hectic (urban), some are understudied and therefore have a high division of labor (rural) (Becher 89).

34 Raymond uses the words "custom," "convention" and "taboo" somewhat indiscriminately. For the purposes of this paper, however, I will call them taboos, because they are less normative rules than prohibitive injunctions.

35 I should clarify here: a piece of Free Software that uses a patented idea without licensing it, is in fact in violation of the patent—and currently the only remedy is to stop using or distributing the software. Free software authors must be very careful not to use patented technology, and so must deal intimately with this conflicting world of idea ownership. In Raymond's article, there is no discussion of the fact that the Noosphere has, so to speak, already been homesteaded by the US Patent and Trademark Office.

36 Forking potentially raises the problem of compatibility and its effect on standardization. This issue is more familiar from Microsoft's Halloween Documents, where the aggressive attempt to "decommoditize" (their word) public protocols in order to make them de facto property of the company is explored. Raymond was responsible for bringing this leaked memo to the attention of the public, it is available with his comments on the Open Source organization website (http://www.opensource.org/halloween/index.html visited 17 April 2001).

37 Greputation, from 'grep' the Unix program that searches for a regular expression. See the Jargon File for further clarification (http://tuxedo.org/jargon/). Greputation suggests that what in speech is accessible only by talking to people face to face, is actually available online as a residue of such discussions—in archives, mailing lists and other openly searchable archives of text. This has led to the research project of Rishab Ayer Ghosh and Vipul Ved Prakash (see Ghosh 1998 and Ghosh 2000) which seeks to measure reputation and contribution to software by explicitly tallying the names, copyrights and email ids in publicly available Free Software packages.

38 Certainly the comparison with scientific dispute resolution is apposite: See again (Merton 1973, Hagstrom 1982, Latour and Woolgar 1979) which are all concerned with what amount to non-formal treaties on the recognition of priority, reputation, scholarly credibility and in the strongest formulation, epistemological claims on truth.

39 Raymond insists on an elaborate genetic explanation for why reputation might have evolved into an incentive structure. However, genetics isn't necessary to explain it, a simpler and more direct explanation is offered by David K. Lewis in Convention (Lewis 1969), which combines insights from analytic philosophy and game theory to describe how conventions arise and stabilize.

40 For those who are not clear on what "speaking" or "online" means: it includes largely written correspondence via email and mailing lists, on public websites, direct written conversation via internet relay chat, or some similar mechanism, and occasionally even face to face contact at meetings, conventions, congresses etc. The greater part of this talk is actually archived somewhere – on servers, mirrors, individual's hard drives – which means it can be searched in the absence of the "speaking" parties, by a third person, or the same people later in time. It changes the meaning of what it means to be "on the record".

41 Even some academics, such as Richard Barbrook, seem quite seduced by this idea: see (Barbrook 1999).
42 For example (Ghosh 1998) explicitly equates reputation with money [cite].
43 Essai sur le don: forme et raison de l'échange dans les sociétés archaïques," first published in 1924.  All citations are from the (Mauss 1990)
44 Potlatch is a massive, violent and agonistic competition between parts of a tribe in which the destruction of enormous wealth determines certain social rankings, cancels debts, and structures commercial activity and economic behavior, among other things.
45 This debate has been repeated in many different contexts where gift-exchanges are observed. Perhaps the most famous readings of this debate are those of Raymond Firth and Marshall Sahlins. Sahlins, in particular, weighs heavy on any discussion of gift-exchange, because his book *Stone Age Economics* has done so much to dispel certain myths about so-called "primitive economies". His reading of Mauss, therefore, is a concerted effort to show that *The Gift* is not at odds with a classical economic understanding—participants in a gift exchange are not altruistic, but just as self-interested as the next economic man.  Unfortunately, Mauss' emphasis on the role of legal obligation is thereby lost (Sahlins 1972).
46 For a more in depth discussion of this, and of its relationship to other theories, such as Marx's notion of fetishism, see the Introduction to (Appadurai 1986).
47 Later, in 1949, Karl Polanyi would describe the same movement as the Great Transformation (Polanyi, 1957), the last gasp of a true laissez Faire market before the 'self-protection' of society overcame it—a  proto-reflexive modernity. In fact, a great deal more historical specificity could be added to the moment when Mauss was writing.  It was in the heart of what was probably the closest the world has ever come to a *laissez-faire* 'free market'—a  market within which prices were determined almost strictly by a price system uncontrolled or manipulated by anything but the traders and their money. Even such an assumption, however, is historically suspect, since this period saw the beginnings of the American Administrative state, the creation of regulatory bodies, the constitutional revolution of the New Deal, the rise of the communist bureaucracies, etc.
48 Though Mauss, like so many of Foucault's precursors, gets not even a passing mention in any of Foucault's work.  See (Foucault 1976) for an elaboration of the "archaeological" method.
49 The connotations of 'technical'—involving machines, inorganic materials, algorithms—tend to blur the sense in which a 'technique' can refer to any kind of know-how—whether embodied in a human or a machine.  Mauss' use of the word captures this in his essay "Les techniques du corps," translated as "Technologies of the body". (Mauss 1950)

50 It is from this footnote that an essay by Jacques Derrida (Derrida 1992, pg. 34ff) considers what it might mean to give away counterfeit money. Derrida assesses Mauss' understanding of the difference between money and gift, while at the same time articulating the madness of trying to maintain a sharp distinction. It does not, however, appreciate Mauss' programmatic innovation on how to conduct a science of exchange.

51 An excellent introduction to the problem is Anne Carson's essay comparing Simonides and Paul Celan, in which she explores how Simondes, as the first poet to be explicitly paid for his work in the context of a Hellenistic patronage system, has been perceived with profound suspicion as a result (Carson, 1999). Uncharacteristically for a poet, Carson also insists on substituting the word "commodity" for "money" when she discusses Mauss.  But nonetheless, the focus of her reading remains the nature of payment, not the alienation of labor.

52 Expectation (*l'attente*) is the word Mauss uses with respect to the origin and function of memory in a short piece on the origin of money (Mauss 1974, p. 106ff).

53 Several well known websites have tried innovating on the control of reputation:  Google.com ranks cites by number of back-links, Slashdot.org uses collaborative filtering systems that allow moderators to control some of the quality of the content, Developers at  Advogato.org have developed an interesting and complex "Trust metric" to fix some of the problems that have arisen at Slashdot.org.  However, the system of reputation *qua* expectation is not confined to this current internet: personal credit ratings, property and asset ownership, savings and earnings, medical and health status, insurability, criminal record, citizenship, etc.  It also concerns the availability of information to individuals themselves: from rapidly updated financial information beamed to your mobile phone, the issuance of quarterly reports, or the wave of Alan Greenspan's hand.. All of these aspects of a person's life and biography form systems for adjusting expectations in the minds and computers of others.

54 See (Levi-Strauss 1987 p. 46-8; Sahlins 1972 p 157ff; and Derrida1992 p. 76-77)  respectively.

55 This is, for accuracy's sake, only true of the GPL, not of BSD style licenses which have only one degree of freedom so to speak, and do not require the subsequent user to grant the same rights. It centralizes the exchange in a way differently than the GPL does.