Freedom Imagined: Morality and Aesthetics in Open Source Software Design

'there is no limit to the harm that proprietary software development can do'
http://www.gnu.org/philosophy/shouldbefree.html

James Leach (with Dawn Nafus & Bernhard Krieger)
University of Aberdeen, Intel Corp, & University of Cambridge

ABSTRACT *This paper is about the interaction between the human imagination and technology among a self-described 'community': that of developers of Free or Open Source Software. I argue that the moral imagination observable in this phenomenon can be understood with reference to its emergence around specific methods of technical production. Principles of openness, truth, freedom and progress, which are also understood as central to the technical production of good software, are reinforced (as a ethical orientation) by their contribution to making 'good' software. A reciprocal dynamic ensues in which better software is seen as dependent on particular social practices and ideologies while these practices and ideologies are given salience by their success in fostering valuable production. Processes key to the generation of this social form are examined before a number of key features of the practice of programming, such as its often combative and individualistic character, and an absence of women in developer communities, are considered in the light of the analysis.*

KEYWORDS *Software, ethics, freedom, progress, social form*

This paper is about an interaction between the human imagination and technology. Its theme is that of how a particular social form comes into being in a dynamic process; one shaped by the production of certain artefacts. The focus is a self-described 'community'[1]: that of developers of a kind of computer software known as Free, or Libre Software, or Open Source Software ('F/LOSS', after Ghosh *et al.* 2002).[2] Although there is a body of literature introducing this phenomenon to which readers may refer (e.g. Ghosh 1998; Raymond 1998; Kelty 2005a), for readers of this journal who may not be familiar with it, I set out some of the basic attributes

and mechanisms of F/LOSS after explaining my methodology. These basic elements are material to my overall argument (and thus part of my ethnography) which is that computer programmers working within this genre share a moral imagination of a particular kind that I illustrate can be understood through reference to its emergence in close relation with specific methods of technical production. The one is reinforced by the other, and a reciprocal dynamic ensues in which better computer software is seen as dependent on particular social practices and ideologies, while certain social practices and ideologies are given particular salience by their success in fostering the production of 'good' code.

Methodology and Approach

The question of how a particular discursive or ideological form comes into being as an aspect of wider relations that include the emergence of material artefacts will clearly involve many factors (cf. Leach 2002). In this instance, the moral imagination in F/LOSS is clearly shaped by (and has come to shape) wider processes in neo-liberal political economy and its forward developments (see e.g. Benkler 2006; Boyle 1996; Lessig 1999; Weber 2004). My interest is in tracing how the moral imagination takes a specific form and has a particular trajectory in F/LOSS because of its precipitation by a series of technological processes. I attempt this without myself becoming invested in the political, social or economic possibilities outlined. That is, I aim to generate a degree of analytic distance from the material in order to gain an understanding of the form from without (Strathern 1999:1–11; Mosse 2006:937).

In a recent article, Coleman and Golub discuss how liberal philosophies informing 'hacker sociality'[3] contain 'multiple, though coherent forms' of discourse on freedom that 'embrace[s] several, sometimes conflicting, historical and present-day moral and political sensibilities concerned with a cluster of commitments' (Coleman & Golub 2008:257). Coleman and Golub invoke what they term a 'cultural sensibility' to refer to this system of linked perspectives and commitments that, while not monolithic or determinant of behaviour, are there to be drawn upon (or reckoned with) by people in particular instances of thought or action. Here I do not replicate that work, although I appreciate their notion of 'cultural sensibilities': my task is not that of discussing the variations between F/LOSS programmers' ideologies and discourses as variations in liberal perspectives, but of describing the way that the moral imagination shapes and is shaped by ways of making technology. This article is intended to complement recent work which discusses the

influence and impact of ideologies around F/LOSS production (Kelty 2008; Coleman 2004; Ghosh 2005; Mackenzie 2007) by emphasising the role that practices in technology production play in shaping emergent social form. Other problems and questions arise around F/LOSS (such as those discussed by the authors cited above), but here I am interested in the *emergence of forms of social action* which are clearly effective within F/LOSS (and beyond). For this reason, I speak of F/LOSS as a social form. I choose not to talk of an ideology or discourse because the focus of my argument and of the phenomenon under scrutiny is on material practices which, as Ratto has persuasively argued, are best understood as simultaneously ideological, discursive and material (Ratto 2005, and see Mackenzie 2007:4–6). The interest here then is not in analysing this community in relation to a particular or general 'tribe' (Raymond 1999) or judging the usefulness of descriptions which relate F/LOSS to gift exchange systems, kinship systems etc. as has been done elsewhere (e.g. Kelty 1999; Zeitlyn 2003). Nor do I attempt to locate variations in the distribution of ideologies. Differences and distributions within F/LOSS may well emerge from the principles I describe, and thus form the subject for further analysis as such. The examples and instances presented here are intended to be exemplary (see Strathern 1992:24–5), rather than representative of various sample populations within the F/LOSS genre. It is in the sense of 'encountering general ideas, values, norms and habits of conduct in particular forms' (Strathern 1992:25) that I use the term 'social form' and my intent is exemplify factors in its emergence around people whose self-definition is as F/LOSS participants.

Given that this is the interest rather than the sociological or historical distribution of the form in any particular group of people, I draw upon a range of different kinds of ethnographic material, ranging from general descriptions of practices in F/LOSS programming to programmatic statements made by the 'elite' of F/LOSS, from material collected from communications over the internet within particular F/LOSS projects (and around them), to scholarly literature on F/LOSS, from my own research with F/LOSS participants, to interviews with those active as developers, including the statements of particular informants who were interviewed by salaried researchers on a project undertaken for the European Commission in which I was Principal Investigator in 2004/5.[4]

This is an attempt (reminiscent of that suggested by philosopher A.N. Whitehead (1929) to describe a generative process rather than a social structure or ideology. The approach picks up the framing of this special

issue. According to Whitehead, the analysis of any entity should focus on the process of its emergence in order not to succumb to an essentialist or a structurally determinist logic. In this case, the way the imagination builds on interactions between machines and persons, and indeed persons in the context of machines demonstrates a process that is neither random, nor determined, but an outcome of specific possibilities made available to the imagination by the activity of engaging in certain technical procedures. The values which emerge and shape the community, while also clearly available to other people, are given a special salience by the practices of writing computer code specifically in the F/LOSS mode.

Some Basics about Computer Software, and about F/LOSS Computer Software

Computer processors operate on the difference between '0's (zeros) and '1's (ones). That is, a pathway for an electrical signal is either to be open or closed, on or off. So instructions for particular tasks the machine is to perform consist of a long series of 0s and 1s in particular configurations. This is called 'machine language' or 'binary code'. At the lowest level of processing all the machine does is 'read', that is, execute, instructions which are given as machine language. Human beings cannot effectively write instructions in such 'language'. There are far too many digits in any one minor instruction when dissolved into its binary elements to make that plausible. Instead computer software programmers write their instructions in human languages (see Mackenzie 2007:24–27). For example an instruction written in one of the many coding languages might be: 'if "c" is true, then execute "d"'. This human readable language is then translated into binary language through what are described as 'lower level' programmes called compilers and translators. These programmes transform each line of the human language into machine-executable binary code. It is the human readable code which is described as the 'source code': the source of the binary code that runs the machine. Free or Open Source software is software which allows a user to see, modify and distribute this source code (DiBona, Ockman & Stone 1999).

When you buy a software programme like Microsoft Word, you receive the binary code which runs on your machine. However, Microsoft Word is what is known in F/LOSS discourse as a 'proprietary' programme. When you buy a piece of 'proprietary software', you get a disk which contains the programme, but which (crucially) is already translated into binary code. Microsoft have to allow the binary code to transfer onto your machine for

it to do anything. They do not 'release', that is, make available to the user/purchaser, the human readable version of that code (source code in human readable language). Users cannot see how it has been written, nor adapt or fix it, should it go wrong or not meet their current requirements.

F/LOSS programmes are supplied to users with the readable code, and thus users can understand how the programme works. They are thus *enabled*[5] to modify that programme, repair it should it malfunction, or adapt it to other uses by modifying the source code and adapting the functions of the programme. This is one distinction between F/LOSS and proprietary software.

F/LOSS developers will tell you that the other distinction is in quality. F/LOSS software, being open to scrutiny, is both more carefully constructed and also continually undergoing improvement by the people who use it. Errors are rapidly and efficiently corrected because of the visibility to users of the source code, and functions are added as part of the process of writing, developing and using the software itself. It is axiomatic that F/LOSS people share source code with one another and examine and modify this source code as part of the process of its use. F/LOSS software projects form self-declared communities that see themselves as collectively providing a better alternative to the way in which software is commercially produced and distributed as a proprietary enterprise.

F/LOSS as Community

F/LOSS members are highly reflexive about their own practices of production and of community. In fact, they have borrowed tropes and metaphors from anthropology in order to describe themselves. One self-conscious perception is that F/LOSS developers tend to differ from 'mainstream' people in their attitudes, ambitions, and *values*, as well of course as in their technical prowess. They are thus described by one well-known participant as a 'tribe' of 'hackers' (Raymond 1999). For F/LOSS people, there is something they call 'hacker culture' (see Kelty 2005a), and descriptions of it draw upon notions of openness, sharing knowledge and information, and a 'gift economy' (op. cit; Kelty 1999) in which (valuable) lines of source code are freely exchanged between people.[6]

Normally we think (correctly) of property as the right to exclude others from using, consuming or benefiting from something. The ownership rights (copyright) the writer has in a piece of F/LOSS software specifically run contrary to this logic. By downloading a F/LOSS programme and running it on a computer, the user agrees that *because* they own copyright over any

modifications they make to the source code, they are in a position whereby they can decide to share those modifications with others on certain conditions. Those are conditions about what the acceptance of the transfer of source code entails for the receiver. Although there have been a number of versions, the General Public License (GPL), written by Richard Stallman of the Free Software Foundation is the original and archetypal instantiation of this principle. It states that by using this (GPL licensed) software, you agree to abide by the terms of the license, which are that users agree to provide their modifications to the code and that they distribute these to others as free software, i.e. under the same GPL agreement, meaning they grant their users access to the source code and the right to change and distribute it as GPL software. Using F/LOSS software thus requires the acceptance of obligations to others discharged through artefactual rather than monetary exchange. This is foundational to participants' understanding of an emergent community.

There are variations on the GPL which allow for more limited control over future use in commercial contexts, but in most cases, modifications are free of price as well as open to scrutiny and development. There are usually a number of people working on the same software programme – developer projects – and they share and distribute their inputs with one another, and indeed the wider world. It is the 'viral' nature of the General Public License which is seen as one of its greatest achievements by participants, 'infecting' any new software that utilises F/LOSS lines of source code, and forcing the developers of that new software to abide by the license and release the programme they have developed as source code for others to subsequently build upon. A common pool of effective source code is one result. 'Community' membership by default (abiding by a delimited and distinct set of obligations to others) is another.

The Value of Freedom

The practice and discourse around the GPL picks up on the notion of freedom; the freedoms specifically built into the GPL (to create, change and modify software, to determine its uses). A better future for all is available as an imagined outcome of writing code under the GPL. As its author writes:

> School should teach students ways of life that will benefit society as a whole. They should promote the use of free software just as they promote recycling. If schools teach students free software, then the students will use free software after they graduate. This will help society as a whole escape from being dominated (and gouged) by megacorporations.[7]

F/LOSS members value their production processes for their open and meritocratic credentials.

> The free software community rejects the 'priesthood of technology', which keeps the general public in ignorance of how technology works; we encourage students of any age and situation to read the source code and learn as much as they want to know.[8]

It is axiomatic that anyone can join the community, and indeed can become significant to software production through developing their skills in writing source code. Rising in influence and prominence in F/LOSS is first and foremost seen as the natural emergence of those with the aptitude and determination to learn to write good code.

F/LOSS is an attractive phenomenon for many people. It is routinely held up as an example of successful production of highly valued software driven by interest and mutual assistance rather than the profit motive; production which makes apparent the potential of, and for, collaboration in the newly networked world (see Ghosh 2005; Benkler 2006). F/LOSS is cited as *the* modern example of productivity outside the frame of conventional intellectual property (see Kelty 2005b), and thus F/LOSS is seen to work against the concentration of cultural, artistic and technical materials in large corporations (e.g. Lessig 2004; Love 2003). It is a source of hope for many, with its model of sharing knowledge objects for the benefit of a wider community spawning significant social movements such as Creative Commons.[9] These movements are explicitly inspired by the ethics built in, as it were, to the production processes and outcomes from F/LOSS.

More widely, literature in Economics discusses F/LOSS in terms of reward and incentive, and the notion of rival and non-rival goods (e.g. Ghosh 1998). Legal scholars discuss developments of property law and the subversion of copyright, the public domain and so on (e.g. Lessig 2004; Boyle 2001, 2005; and see Vaidhyanathan 2001, 2006), others focus on the efficiency of distributed working and the organisational aspects of information production (e.g. Agrain 2005; Benkler 2001, 2005). Then there are the struggles against large corporations, the freedom fighter image of a few techno-literates changing the way the economy works, challenging large capital, and winning (see Coleman & Golub 2008:272).

It is undeniable that F/LOSS programmes are highly successful as computer software. Most world wide web servers run on 'Apache', a large F/LOSS project. The F/LOSS computer operating system Linux runs much contemporary

commercial networked computing. Governments around the globe (including the US) are migrating their systems to Linux because of its usability and transparency. In fact, the place where F/LOSS is least visible, unless you are an initiate, is on your personal computer desktop.

The F/LOSS community borrows the language of the anthropologist and the concept of gift economy to highlight their differences from the standard practices of the commodity economy in which they successfully operate. Tropes of progress, and a fairer, more democratic, and safer future appear continually as aspects of their self description. I now set out to show that the 'evolution' of programmes themselves in F/LOSS is considered as a 'natural' process of improvement. This allows both a notion of a 'natural' emergence of hierarchy based on competence, and secondarily encourages an elision of the moral and descriptive language used of F/LOSS.

Languages, Judgements, and Establishing Truth

Up to now, I have set out to show how the moral values of community and progress are embedded in descriptive language in and around F/LOSS and how in turn this builds on a sense of the moral agency of F/LOSS software as a system of production itself. The technical/organisational procedures, in other words, precipitate the elision of the practical with the moral. The process of programming works as a technology of the imagination because it instantiates a connection with the moral principles outlined by its advocates, and supports the 'naturalisation' of a moralised discourse.

I now aim to show how the process whereby this elision is facilitated is an aspect of writing code itself in the F/LOSS manner. I seek to establish three points. Firstly, that the binary nature of machine language has an imaginative effect on perceptions of right and wrong for coders. Secondly that the moral discourse is wrapped up in an aesthetics of code. Judgements made about code are seen as aesthetic judgements which nevertheless are seen to have an objective basis in reality (moral judgements and truth claims overlap). Finally, I will argue that this aesthetics of code generates a conception of future potential that in turn makes the activity of writing code a version of exploring and opening up physical and mental frontiers.

As described at the outset, computers are machines which process a long series of instructions of a binary nature: on and off, yes or no. There are two metonymical extensions of this principle among participants which are important: Firstly, that at this level of simplicity, humans can act directly upon the world external to them. Acts which operate on the simple principle of

true or false are thus more basic than subsequent elaborations that might be possible upon them. Secondly, that there are, at base, true and untrue statements. This is a 'binary' approach to judgement. The external world ('reality') is an important concept here. What I aim go give a sense of in the following description is how a particular version of 'reality' both comes into being, and becomes the basis for action. The logic runs something like this: machines are (essential) tools with which humans manipulate external reality. The primary concern is to make the tool work; that is, make a tool which has the desired effect in the external world. And thus people make judgements about computer code on the basis of how well and efficiently it works, having in mind that efficient working is because, in some sense, the code equates more closely with effects in the external world upon which the machine operates. Importantly, these judgements of fitness or effectiveness merge with more subjective aesthetic judgements.

Truth statements ('if "x" is true, then y') are a central feature of programming languages. The software functions if the truth statements are true. Often making software 'work' is a zero sum situation; it either runs or fails to run, and F/LOSS programmers describe the exhilaration of finally making something 'work'. As if this aspect of code served as a wider metaphor, judgements can be – and are – made on the basis of what counts as 'good code' and 'bad code' as if this were an objective matter. This sets the tone of much online discussion in and around software projects. One informant, for example, reported an instance where a developer was not able to convince his colleagues about the worth of his code due to his limited English language skills (which was the working language for organising the relations of production in the project). He managed it, however, by claiming he would let the code speak for its superiority itself.

There is also a traceable fusion of ideas about good and bad code, and morality and aesthetics, in arguments over which of the multitude of programming languages is 'best'. What constitutes 'good' or 'bad' in code seems to depend upon whether F/LOSS people are talking about Perl, C, Lisp, Java, Python etc. (to cite just some of the many programming languages). In these debates, logical arguments are interspersed with appeals to aesthetic criteria and to ethical/political and economic values. For example in a recent book by a prominent advocate for F/LOSS, and within F/LOSS of the programming language 'Perl', Clay Shirky writes:

> Perl is a viable programming language today because millions of people woke up today loving Perl and, more important, loving each other in the context of Perl. Members of the community listen to each other's problems and offer answers as a way of taking care of one another… [C]ommunal interest turned out to be a better predictor of longevity than commercial structure… the question 'Do the people who like it take care of each other?' turns out to be a better predictor of success than 'What's the business model?' As the rest of the world gets access to the tools once reserved for the techies, that pattern is appearing everywhere, and it is changing society as it does (2008:257–259).

The process of establishing truth can be illustrated with reference to the practice of 'flaming' (acrimonious exchanges of messages). Although it is considered ideal that good code and good languages will speak for themselves, evaluations are often strongly contested. This practice is consciously endorsed by members of F/LOSS. For example, Linus Torvalds, the influential leader of the Linux Kernel project, recently argued against the utility and value of the approach and programming language used in a project called 'Subversion'.[10] A key developer on Subversion called Torvalds' presentation 'rude almost beyond belief', however, he acknowledged that 'some of his technical points were valid, and the Subversion developer community has been good about separating the style from the substance'.[11] Later Torvalds explained what he likes about 'flaming' in reference to this exchange:

> I like making strong statements, because I find the discussion interesting. In other words, I actually tend to 'like' arguing. Not mindlessly, but I certainly tend to prefer the discussion a bit more heated, and not just entirely platonic. And making strong arguments occasionally ends up resulting in a very valid rebuttal, and then I'll happily say: 'Oh, ok, you're right'.[12]

Developers then, on occasion, vociferously defend their work or proposals. In doing so, they demonstrate knowledge and establish what good coding is. These debates and often heated arguments take place in electronic chat rooms or over email lists which are open to all members of a project. They are thus almost public tournaments ('flame wars') which may continue for days or weeks.

If coding alludes to knowledge, flaming asserts it in no uncertain terms. While there is scholarship that argues that online communication lends itself to flaming (Scott, Semmens & Willoughby 2001; Michaelson & Phol 2001; Herring 1996), in F/LOSS it appears from informants that it is particularly rewarded. Many informants then explain their behaviour as a way to elicit the objective truth.

Aesthetics and Truth

What emerges is that knowledge is ordered in F/LOSS processes by two separate, but related, dynamics. One is that F/LOSS assumes the existence of real and true knowledge which is discoverable. The second is that the aspect of this is the importance of making tools that work on this external reality; that is, software programmes 'sit' in this externalised world; it is their function to achieve ends in that externalized world. Value claims here are first and foremost truth claims. The tools and made objects should be determined in their form by the reality that they are designed to operate upon. Truth also has the characteristic of simplicity. People in the F/LOSS community speak continually of the elegance and simplicity of good code. Experts can 'see' good code just by looking at the source language. Redundancy and complication are rejected in this aesthetic of effectiveness.

A central component of the F/LOSS ethos is that working openly and sharing the source code of software enables improvements to evolve more effectively, and that as a whole 'better' software is produced ('Free software has developed such practical advantages that users are flocking to it for purely technical reasons' (Stallman 1999:69). The concept of 'better software' (a material judgement), which arises from 'better' processes of production (a moral judgement) conceals another complex series of understandings and judgements generated by familiarity with and proximity to the workings of the machine (computer) itself.

An ethos of expanding the boundaries of knowledge and making code as functional as possible stands as its own moral good. Source code that is written in a F/LOSS manner has these potential qualities for participants. Above all, code must evolve ('Software development used to be an evolutionary process' [before proprietary software firms took control], which was 'a more efficient system'.[13] A central and common component of the F/LOSS ethos is that working openly and sharing code enables evolution to happen more effectively, and that as a whole 'better' code is produced. Language of the natural (evolutionary process) comes to stand as evidence for the correctness of this way of proceeding, and thus making a better future.

Code components that are built beautifully and cleverly establish a future *potential* by establishing the validity of certain truth claims, and this in turn makes it possible for others to identify possible future routes for developments in software. Particularly effective pieces of code are called 'Good Things': the described code is self-evidently worthy. Whereas Good Things are treated as discrete entities capable of standing for themselves, the knowledge (truth) they

reveal is considered to be transferable to other software ends and allow one to create nearly anything. Because code both enacts knowledge and makes knowledge evident, in an illocutionary way it reproduces the imagined space of a frontier by revealing the truth of what exists and thus new potentialities. Potential future directions based on past development are made convincing by embedding them in 'hard', 'material' substance: the machine.

The commitment to this frontier permeates the everyday practices of both programming in the F/LOSS mode and creating a F/LOSS community. It sets the conditions on which reputations are made, prestige is gained, and software is designed. It matters not just what sort of software is being developed – i.e., what task the software performs – but how the code achieves its goal, how that demonstrates the fitness of the language used etc.

The beauty of code comes to be an aim in itself in this overlap of practical and moral truth. Often informants claimed that some proprietary software projects have not yet opened up their source code because of its messiness; that is, proprietary software producers would not dare to open source their code because they were ashamed of how its functionality has been achieved. The knowledge embodied in and revealed by code are descriptive creations to them, in other words. They describe a reality (functionality) that is external, and at the same time the code acts in that external reality: it functions. The importance of how code works articulates the frontier of knowledge just as much as, if not more than, the resultant task made possible by the new software.

The ideal of pushing knowledge forever forward also features in the debates surrounding the usability (or otherwise) of open source software. For instance, employing unstable versions of software is considered to demonstrate technical competence. It serves as evidence that the author is doing new exciting work, and it also enables users to develop their knowledge to adjust or expand the software as appropriate. Because the source code is available at the point of use, users have much more autonomy and control over the software versions sitting on their personal machines. Whereas Neff and Stark (2003) have described the constant state of flux in proprietary software as 'permanently beta' ('beta' referring to a publicly-released test version of software), in F/LOSS the situation is acute. This makes certain demands on and assumptions about users as it is far more difficult to install and use beta versions of software. The binary judgements of good and bad that are the building blocks for an entire aesthetic of coding result in a judgement of quality as an equivalent to code's value for potential future work.

The relation between aesthetic forms and moral principles is a well established area of interest and debate in the history of European and American thought. The consequence of the F/LOSS aesthetics of code appear in how judgements are made about what code is good, and that equates to a sense of the good code as more real, that is, more effective in achieving its ends without a waste of resources, without inconsistency and so forth. Reality is not just effects, but particular effects which equate to moral principles as well. Aesthetics and morality then combine to produce a powerful sense that F/LOSS code can have qualities which make it more fit for use, more effective in achieving externalised ends, and thus better than other forms of software.

As the ends to which computer code addresses itself are vitally important: communication across the globe, sharing information and knowledge, techno-medical interventions (and so on) there is also another moral imperative to producing good code – it enhances the possibilities of human advancement. 'Good Things', things that work without redundancy, are both aesthetically pleasing, and morally positive in what they enable others to do. They also have the potential to reveal the structure of future reality. Readers can access data supporting these assertions quite directly. The hugely successful F/LOSS web browser Firefox has easily accessibly links to 'evangelism' advocates and advice for participation http://www.firefoxevangelism.com. US College students are encouraged to become 'Campus Reps' for these programmes, advocating on the basis of their superiority in more than just technical terms ('Tell everyone you know why you think Firefox is the browser choice of your generation. Is Firefox prominent on campus? Help make it so!')[14]

Autonomy and Progress

The 'free' in free software is envisioned as part of a broader ethos of freedom of speech and volition rather than as a reference to (zero) price. The key notion is free as in 'free speech', not as in 'free beer' (Free Software Foundation 1996). French language here makes the distinction more accurately – *libre* rather than *gratis*. Promoters of *logiciel libre* (free software) in France, refer to ethics as necessary for participation. Indeed, the term 'open source' tends to be used in commercial contexts (thus avoiding the problematic associations in English that 'free' has for commercial activities), and free or *libre* in public advocacy modes. While some developers are less interested in these distinctions than others, they do share a consuming enthusiasm for the idea that coding is its own moral reward.

In F/LOSS moral language is used as if it were straightforward description. For example, 'transparency' is what F/LOSS stands for. The enclosure of an intellectual commons is its opposite. So other social movements feed from the model of F/LOSS, drawing on a variety of notions which carry moral implications in the face of the alienating, atomising and greed-inducing processes of corporate expansion and appropriation. The gift economy (see Kelty 1999) description is another such trope, suggesting as it does, relations of kinship (Zeitlyn 2003) and community. Freedom, as an imagined property of 'gift economies', is seen as central, with an extremely high value placed on individual autonomy and the ways that autonomy is central to people's participation in F/LOSS, and in turn, how its acceptance and validity can be demonstrated and enhanced by F/LOSS production.

> The idea that the proprietary software social system – the system that says you are not allowed to share or change software – is antisocial, that is unethical, that it is simply wrong, may come as a surprise to some readers. But what else could we say about a system based on dividing the public and keeping users helpless? (Stallman 1999:54).

The model has had great purchase then beyond the actual development of code in part because the ethos and approach to this technical activity draws upon well established and highly valued principles of individual autonomy, of meritocracy, and of progress. But what F/LOSS has achieved for its participants and supporters is that the realisation of such ideals seems attainable through the specific process of writing and circulating source code.

Individual Moral Frontiers

'Better' code demands continual improvement, and F/LOSS programmers concieve of themselves as working on a technological (and generative/creative) frontier (cf. Helmreich 1996). Being on the cutting edge of technology serves as is its own goal for many. This process has been likened to craft production (Coleman 2001) in that members often describe their motivation to work in projects as providing the reward of problem solving, 'scratching an itch' by producing a tangible solution to a defined problem (Raymond 1999).

However the craft in question is satisfying only when it is seen in terms of a model of knowledge that locates agency and activity within the technologist, who thereby himself pushes back the frontier of knowledge. Here we see the personal and the moral interlink in the activity of writing F/LOSS code. Pursuing one's 'craft' is also to behave ethically and to have a positive

influence in society. The moral is presented as if it were merely an outcome of writing source code and fulfilling one's personal ambitions.

On this frontier of knowledge, tradition and repetition have no place other than as building blocks upon which to take one's own work further. Ullman (1997) describes working as a programmer as a process of sitting at the edge of fractal knowledge. There is always more to know, and there will always be more languages and technologies available than any one person can know. She thinks certainty is important in such an enterprise: one has to be quite certain that one's knowledge is 'enough', even though it almost never is. No one can claim absolute expertise in such a wide and rapidly developing domain (see also Downey 1998).

I pointed to an elision of moral and descriptive language in and around F/LOSS. And it is here that the imagination plays such a significant role. For F/LOSS participants, writing source code is progressive. It allows the human race development it otherwise would not have, such as increased control of the environment, better communication etc. There is then a wider purpose to coding. It has moral weight above and beyond the desires of the programmer to problem solve or be creative for their own satisfaction. Ethics and the imagination of a better future merge here into an unquestioned 'righteous' quality to F/LOSS software. Ethics and politics are pursued through making material objects which then in themselves, without the need for 'social' action, achieve political ends.

Conclusion: Hierarchy and Gender

The paper so far has been based on an analytic premise. I have been concerned to trace the process of the emergence of ideas of freedom and its value as aspects of an emergent social form which includes the production of effective technologies. Having done this, I am now going to turn briefly to look at one aspect of the sociological distribution of these forms. This is *not* to extend my argument, but to demonstrate the power of the inter-relation of technical processes and ideas that I have described. The efficiency with which these procedures have shaped the imagination is perhaps most clearly shown by certain aspects of F/LOSS in which their precedence over other considerations is dramatically demonstrated. That is, other values, while available to participants, and endorsed by them, are not congruent with the technical organisation of F/LOSS, and are therefore elided.

There are some arresting aspects of F/LOSS, if viewed as a social movement. The most striking of which is a potential contrast between a central

ethic of freedom and meritocracy, and an extremely low participation of women. In fact, (Ghosh *et al.* 2002) discovered through extensive surveys and questionnaires among the community that less than 2 percent of active developers were women in 2002. In certain spheres of contemporary life gender exclusion is accepted, sports teams being an obvious example in Europe and America. So there is nothing terribly surprising about the number of women participants *of itself.* However, given the ideology of freedom which is central to the formation of these groups, and the contexts in which F/LOSS is written, it does seem to require sociological explanation.[15] While the numbers of women in computer science generally is lower than that of men (with 72 percent of coders who work in proprietary software contexts being male (National Science Foundation 2004) the figures in F/LOSS are remarkable. Another aspect one might find surprising on second glance is that there is an extremely hierarchical system of authority which operates in F/LOSS projects, and moreover, that authority is often enforced in displays of aggressive argument and belittling of others, as discussed above.

It is important that readers understand that I neither aim to criticise, nor to excuse, what occurs in the production of F/LOSS. My aim has been a description of how and why the community of developers comes to take the social form that it does. That description inevitably involves perceptions of 'the social' among F/LOSS participants as well as of the machine, of code, and of material effect. For here we see a clear example of a social creation in a holistic sense. Morality and ethics are inseparable from the objects which the community produces. Politics, and an imagined future, are pursued through the construction and development of software, of objects, which themselves are to carry the burden of, and are believed to instantiate, an ethical and moral vision. Politics in this case can be engaged though writing software in a particular manner. 'Computer users should be free to modify programs to fit their needs, and free to share software, because helping other people is the basis of society.' (Stallman 1999:55). While commonly perceived as technically more efficacious, F/LOSS also has, 'social advantage . . . and an ethical advantage, respecting users freedom.' (ibid.:61).

In this self-reflection and description, and indeed in the at present small, but now rapidly growing scholarly literature around F/LOSS, some aspects of the social reality of F/LOSS are absent. Gender has not figured in a serious way.[16] I say 'serious' consciously. Part of the self-descriptions one often meets among F/LOSS programmers is of awkwardness in social situations. Concerns about how to attract the opposite sex abound in the narratives and communications

between members of the community. They arise specifically because of a self-declared 'culture' around 'computer geeks' emphasising the importance of long working hours, familiarity with esoteric and alternative languages, and a dedication to technical tasks which over-ride other considerations.

As Nafus, Krieger and I have suggested, this may be no co-incidence. The fact that certain aspects of the constitution of the social form of F/LOSS do not come under scrutiny internally is a direct effect of the power of the imagination in relation to software's potential contribution to a better world. And that is inextricably bound to the engagement with the material world, the reality of the machine, and a faith that digital technology will structure our future social and cultural existence (E. Leach 1968; J. Leach 2005). As Kelty correctly observes, F/LOSS 'is distinguished from other forms and practices of software production for many reasons, but most interestingly because its practitioners discuss it not simply in technical terms, but as a philosophy, a politics, a critique, a social movement, a revolution, or even a "way of life"... It seems to offer an answer to the 21st century question of how we should live' (Kelty 2004:499). Given these 'indigenous' claims, and indeed the undeniable force of the phenomena as both model for collaborative work and producer of software it is perhaps *not* strange that aspects of the relations of production of the code itself, and how this articulates with the wider role of gendered positions in technological and IT arenas, has until now been absent.

How do I come to this conclusion? There are several factors at work. The first is that F/LOSS participants see themselves as operating on a frontier, with the future potential of well-written code prominent in its valuation. It opens new horizons, provides new routes 'forward'. At a wider scale, the notion of making better software itself positions coders at the edge of what is possible, driving human advancement. With this level of importance attached to the products of F/LOSS, other kinds of consideration take second place. The inseparability of code from valued moral and political principles reciprocally constitutes the production of code as not only an instantiation of correct action, but the very materiality of code itself is a proof of the truth and fitness of these principles. It is here that metonymic extensions of the functioning of a technology become 'technologies' in their own right: technologies in which imaginative constructions of the future are seen as plausible because they are extensions of the very processes whereby functional computer code is produced. It is in the confluence of notions of the real, the aesthetic, the good and the true that the imaginative power of F/LOSS is most clearly seen.

I have described how the binary character of machine language, and the thinking necessary to making things work in practice encourages an emphasis on truth and falsity in the moral and social world as well as the technological. I have discussed how the 'openness' of F/LOSS software precipitates notions of the value of freedom and egalitarian community, and I have demonstrated that the potential for innovation and development in the software renders the activity as one of 'frontier exploration' which clearly has analogies in how the social movement of F/LOSS is also seen. It is transformative of *society*. These all have consequences for the emergent social form which is F/LOSS, with little room for either social niceties in establishing truth, or a mechanism for considering the make-up of the community. To do so would potentially work against the values of freedom, autonomy and progress as they are constituted in F/LOSS and the imaginative technology which demonstrates to participants that these good things can be advanced by making better software. Making good software takes precedence over other objectives *because* it is the way to achieve them.

Acknowledgments

I am grateful to a group of F/LOSS participants who engaged in constructive dialogue of the issues around gender and F/LOSS during a workshop in Cambridge in 2006 (see Nafus, Leach & Krieger 2006) and in particular, to Hannah Wallach who was both a facilitator for that workshop and significant interlocutor for the F/LOSSPOLS project. I also thank Geoffrey Lloyd and Finn Brunton, the editors of this Special Issue, and the three anonymous *Ethnos* reviewers who have all generously contributed to the paper.

Notes

1. From here on, I adopt the term used by F/LOSS programmers themselves to describe the people involved in the production of this software: a 'community'. I am not making a sociological claim that F/LOSS is a community, but making an observation that the idea that F/LOSS is a community is an ideological component of the social form that is F/LOSS. That has effects. These may indeed be the creation of an actual community, but the mechanisms of that are not my concern in this paper.
2. F/LOSS: an acronym for Free/Libre/Open Source Software, and has the advantage for an outsider of glossing over distinctions between coders and coding projects based on the community's internal, and shifting, (political) divisions. There are many published accounts of the differences between Free Software ideologies, and Open Source ideologies (e.g. DiBona *et al.* 1999) but these are not discussed here. Instead, I undertake an analysis of the generation of an encompassing social form; the outcome of principles and assumptions which help to form the social interactions through which F/LOSS software is written.
3. In this context, 'hacker' carries only the connotation of a high level of ability with writing and understanding the languages in which computer code is written, not someone with criminal intent.

4. Bernhard Krieger and Dawn Nafus. See www.flosspols.org.
5. To (consciously) highlight a term used in F/LOSS.
6. For a discussion and analysis of the use of the notion of the (Maussian) Gift among F/LOSS participants, and their investment in the notion of there being 'hacker culture' see Kelty (1999, 2004).
7. www.gnu.org/philosophy/schools.html.
8. www.gnu.org/philosophy/schools.html.
9. www.creativecommons.org.
10. http://codicesoftware.blogspot.com/2007/05/linus-torvalds-on-git-and-scm-html.
11. http://radar.oreilly.com/2007/07/why-congress-needs-a-version-c.html.
12. www.efytimes.com/efytimes/21160/news.htm.
13. Richard Stallman. www.gnu.org/philosophy/shouldbefree.html.
14. http://www.spreadfirefox.com/campusreps.
15. Having dwelt on the way discourses around these technologies draw upon 'natural' or 'scientific' theories of human action and potential in establishing their social form, I do not here consider the commonly heard recourse to theories of 'natural' differences between male and female brains to explain the low participation of women. Such theories (which have been present in the ethnography that I have collected) must be taken as ethnographic data rather than analytic end points.
16. Lin 2005 being an exception.

References

Benkler, Yochai. 2001. Intellectual Property and the Organisation of Information Production. *International Review of Law and Economics,* 22(1):81–107.

—. 2005. Coase's Penguin, or, Linux and the Nature of the Firm. In *CODE*: *Collaborative Ownership and the Digital Economy,* edited by R. Ghosh. Cambridge MA: The MIT Press.

—. 2006. *The Wealth of Networks*. New Haven: Yale University Press.

Boyle, James. 1997. *Shamans, Software and Spleens: Law and the Construction of the Information Society*. Cambridge MA: Harvard University Press.

—. 2001. The Second Enclosure Movement and the Construction of the Public Domain. www.law.duke.edu/pd/papers/boyle.pdf.

Coleman, E. Gabriella. 2001. High Tech Guilds in the Era of Global Capital. *Anthropology of Work Review,* 22(1):28–32.

—. 2004. The Political Agnosticism of Free and Open Source Software and the Inadvertent Politics of Contrast. *Anthropological Quarterly*, 77(3):507–19.

Coleman, E. Gabriella & Alex Golub. 2008. Hacker Practice: Moral Genres and the Cultural Articulation of Libersalism. *Anthropological Theory*, 8(3):255–277.

Comino, Stefano., Fabio M. Maneti & Maria L. Parisi. 2005. From Planning to Mature: On the Determinants of Open Source Take Off. http://opensource.mit.edu/papers/Comino_Maneti_Parisi.pdf.

DiBona, Chris., Sam Ockman & Mark Stone (eds). 1999. *Open Sources: Voices from the Open Source Revolution*. Sebastopol, CA: O'Reilly & Associates.

Downey, Gary L. 1998. *The Machine in Me: An Anthropologist Sits Among Computer Engineers*. London: Routledge.

Free Software Foundation. 1996. *The Free Software Definition*: Free Software Foundation.

Ghosh, Rishab A. 1998. Cooking Pot Markets: An Economic Model for the Trade in Free Goods and Services on the Internet. *First Monday* 3(3). www.firstmonday.org.issues/issue3_3/ghosh/index.html.
—. (ed.). 2005. *CODE: Collaborative Ownership and the Digital Economy*. Cambridge MA: The MIT Press.
Ghosh, Rishab A., Rüdiger Glott, Bernhard Krieger & Gregorio Robles. 2002. *Free/Libre and Open Source Software: Survey and Study. Part IV: Survey of Developers*. International Institute of Infonomics/Merit.
Helmreich, Stefan. 1996. *Silicon Second Nature: Culturing Artificial Life in a Digital World*. Berkeley: University of California Press.
Kelty, Christopher. 1999. *Hau* to do things with words. www.kelty.org/or/index.html.
—. 2004. Culture's Open Sources: Software, Copyright, and Cultural Critique. *Anthropological Quarterly* 77:499–506.
—. 2005a. Geeks, Social Imaginaries, and Recursive Publics. *Cultural Anthropology*, 20(2): 185–214.
—. 2005b. Trust among the Algorithms: Ownership, Identity and the Collaborative Stewardship of Information. In *CODE: Collaborative Ownership and the Digital Economy,* edited by R. Ghosh. Cambridge MA: The MIT Press.
—. 2008. *Two Bits: The Cultural Significance of Free Software*. Durham, NC: Duke University Press.
Latour, Bruno. 1999. *Pandora's Hope: Essays on the Reality of Science Studies*. Cambridge, Mass.: Harvard University Press.
Leach, Edmund. 1968. *A Runaway World? The 1967 Reith Lectures*. London: British Broadcasting Corporation.
Leach, James. 2002. Drum and Voice: Aesthetics and Social Process on the Rai Coast of Papua New Guinea. *Journal of the Royal Anthropological Institute* (N.S.), 8(4):713–734.
—. 2005. Being in Between: Art-Science Collaborations and a Technological Culture. *Social Analysis,* 49(1):141–160.
Lessig, Lawrence. 1999. *Code and Other Laws of Cyberspace*. New York: Basic Books.
—. 2004. *Free Culture: How Big Media Uses Technology and the Law to Lock Down Culture and Control Creativity*. New York: Penguin.
Lin, Yuwei. 2005. A Techno-Feminist Perspective on Free/Libre/Open Source Development. http://opensource.mit.edu/papers/lin5.pdf.
Love, James. 2003. Prescription for Pain. In *Le Monde Diplomatique*. Paris.
Mackenzie, Adrian. 2007. *Cutting Code: Software and Sociality*. New York: Peter Lang.
Michaelson, Greg & Margit Phol. 2001. Gender in E-mail-based Co-operative Problem Solving. In *Virtual Gender: Technology, Consumption, Identity,* edited by E. Green & A. Adam. London: Routledge.
Mosse, David. 2006. Anti-social Anthropology? Objectivity, Objection and the Ethnography of Public Policy and Professional Communities. *Journal of the Royal Anthropological Institute* (*N.S.*), 12:935–956.
Nafus, D., J. Leach & B. Krieger. 2006. *Free/Libre/OpenSource Software Policy Support: Gender Track*. European Commission.
Nafus, D., B. Krieger & J. Leach. n.d. Patches Don't Have Gender.
National Science Foundation. 2004. Women, Minorities, and Persons with Disabilities in Science and Engineering. In *NSF 04-317*. Arlington, VA: NSF.

Neff, Gina & David Stark. 2003. Permanently Beta: Responsive Organization in the Internet Era. In *The Internet and American Life,* edited by Philip Howard & Steve Jones. Thousand Oaks, CA: Sage.

Ratto, Matt. 2003. The Pressure of Openness: The Hybrid Work of Linux Free/Open Source Kernel Developers.

—. 2005. Embedded Technical Expression: Code and the Leveraging of Functionality. *The Information Society,* 21(3):1–24.

Raymond, E. 1998. The Cathedral and the Bazaar. *First Monday,* 3(3). http://www.firstmonday.org/issues/issue3_3/raymond/

—. 1999. The Revenge of the Hacker. In *Opensources: Voices from the Open Source Revolution,* edited by Chris DiBona, Sam Ockman & Mark Stone. Sebastopol, CA: O'Reilly and Associates.

Scott, A., L. Semmens & R. Willoughby. 2001. Women and the Internet: The Natural History of a Research Project. In *Virtual Gender: Technology, Consumption and Identity* (eds) E. Green & A. Adam. London: Routledge.

Stallman, Richard. 1999. The GNU Operating System and the Free Software Movement. In *Open Sources: Voices from the Open Source Revolution,* edited by Chris DiBona, Sam Ockman & Mark Stone. Sebastopol, CA: O'Reilly Associates.

Strathern, Marilyn. 1992. *After Nature: English Kinship in the Late Twentieth Century.* Cambridge: Cambridge University Press.

—. 1999. *Property, Substance and Effect: Anthropological Essays on Persons and Things.* London: Athalone Press.

Turkle, Sherry. 1998. Computational Reticence: Why Women Fear the Intimate Machine. In *Sex/Machine: Readings in Culture, Gender and Technology,* edited by P. Hopkins. Bloomington: Indiana University Press.

Ullman, Ellen. 1997. *Close to the Machine: Technophilia and its Discontents.* San Francisco: City Lights Books.

Vaidhyanathan, Siva. 2001. *Copyrights and Copywrongs: The Rise of Intellectual Property and How It Threatens Creativity.* New York: New York University Press.

—. 2006. Afterword: Critical Information Studies: A Bibliographic Manifesto. *Cultural Studies,* 20(2–3):292–315.

Weber, Steven. 2004. *The Success of Open Source.* Cambridge M.A.: Harvard University Press.

Whitehead, A.N. 1929. *Process and Reality.* New York: Macmillan.

Zeitlyn, David. 2003. Gift Economies in the Development of Open Source Software: Anthropological Reflections. *Research Policy,* 32(7):1287–1291.