David M. Berry

THE CONTESTATION OF CODE

A preliminary investigation into the discourse of the free/libre and open source movements

This paper uses discourse analysis to examine the free/libre and open source movements. It analyses how they fix elements within the order of discourse of computer code production. It attempts to uncover the key signifiers in their discourses and trace linkages between the sedimented discourses of wider society. Using discourse theory and critical discourse analysis, the theoretical foundations underpinning each of the movements are critically examined and the effect on the wider developer and Internet community is discussed. Additionally, this paper seeks to recommend discursive strategies that could be employed to avoid the threat of colonization by neoliberal discourse and the consequent challenge this has for the ideas of freedom, liberty and community within the developer communities' own discourses.

Keywords digital media; cyberspace; GNU; GPL; discourse; Internet; open source; free software; discourse analysis

> While free software by any other name would give you the same freedom, it makes a big difference which name we use: different words convey different ideas.
>
> (Stallman, 2003e)

> The term "free" software is very ambiguous (something the Free Software Foundation's propaganda had to wrestle with constantly). Does "free" mean "no money charged?" or does it mean "free to be modified by anyone", or something else?
>
> (Raymond, 2001c)

The Free Software Foundation (FSF) and the Open Source Movement (OSM) are engaged in a struggle at the level of discourse. The two Internet-based movements attempt to fix the polysemic elements within an order of discourse[1] surrounding the production and interpretation of computer based programming code[2] in order to establish a closure (Laclau & Mouffe, 2001, p. 110; Phillips & Jørgensen, 2002,

Critical Discourse Studies Vol. 1, No. 1 April 2004, pp. 65–89
ISSN 1740-5904 print/ISSN 1740-5912 online
http://www.tandf.co.uk/journals DOI: 10.1080/17405900410001674524

Routledge
Taylor & Francis Group

p. 28). This paper examines which of the two movements is more likely to achieve closure, why it might be successful, and the larger social and political implications of this struggle.

Key research questions are: (1) Why should the construction of computer code be a subject worthy of social and political research; and (2) Why is it an important topic?

In seeking to answer these questions a preliminary coding is made of a number of OSM and FSF documents and interviews. These are then examined using a comparative approach to uncover meaning and contested concepts by contrasting the way the two movements use key terms. This paper is intended to contribute to ongoing broader research into the free/libre software and open source movements.

Although the order of discourse for computer code may seem somewhat esoteric, a broader political project appears to be manifested within the debates between the two movements (Feenberg, 1995; Lessig, 1999, 2002; Moglen, 1999; Raymond, 2003a; Stallman, 2002a). In short, this paper argues that these movements crystallize discursively a more substantive challenge for wider society, namely issues surrounding the legitimacy of technocratic society, reflexive modernization, the democratization of technology, and the public deliberation of technology policy. The respective positions of the two groups provide a unique case study for theories about modernity and technology and raise questions for further research (Berry, 2003, 2004; Berry & Moss, 2004).

This paper uses a combination of Laclau and Mouffe's discourse theory (Laclau & Mouffe, 2001; Phillips & Jørgensen, 2002) together with elements of Fairclough's critical discourse analysis (Fairclough, 1992) to analyse both the contents of texts and how wider sedimented hegemonic discourses within society may intervene to suppress discursive conflict within this order of discourse.[3] This, it will be argued, may lead to a naturalization of the OSM's order of discourse (Fairclough, 1992).

First, a brief outline and definition of source code is given. An overview of both the requirement and procedure for the creation of source code and key analytical categories used in this paper will be outlined. Second, an examination is made of elements and key signifiers in discourses from the FSF and OSM respectively. Subject positions will also be presented from each discourse. Third, I offer a comparative discursive analysis, which is applied to the results of the discourses' key signifiers. In the final section, these streams are drawn together and the paper outlines possible issues for the FSF and the likely naturalization of the OSM's discourse (Fairclough, 1995). In addition, I will examine possible alternative discursive strategies and possible avenues for future research.

Source code

Computer code is necessary for the operation and control of all technological machinery relying on microprocessors. This code is the concretization of general algorithms instantiated into particular programming languages. For example, the mechanical process required to move a unit of data from point A to point B could

be written in English. However, for the computer to execute this command, this algorithm needs to be translated into machine code (Stallman, 2002a).[4] At this level the code is represented digitally and numerically and is very difficult to write directly. Indeed, the mythology of expert programmers and hackers[5] dates back to the time when this was the only means of programming computers (Post, 2003; Williams, 2002).

The production of computer code at this low level is complex, time-consuming and slow. To enable programs[6] to be written more quickly and remove complexity, an abstraction of the underlying machine code is used instead. These are the contemporary programming languages, often known as third generation languages (3GL), in which the human programmer or coder is usually required to write, for example, C++, Java, and Basic. These highly abstracted languages use a formalized syntax and are usually constructed around simplified English keywords. Together with symbols and punctuation, programs are written in a structured syntactical style made up of statements, loops and conditionals to construct the logical operation of the program.

Programs are written in preliminary documents, usually plain text files, which contain the logic of program operation and are known as source code files. In addition to the concrete controlling logic of the program flow, the source code will often contain a commentary by the programmer in a special textual area usually delimitated by special characters, for example, "REM" in Basic. These comments assist both the programmer and others wishing to understand the programming code.[7] Additionally, these textual areas are used to demonstrate authorship, list collaborators and document changes.

For source code to be usable, it must be transformed using a compiler[8] into the machine code that can be read and executed by the computer, known as the "object file" or "executable". This process removes the original source code files after the executable has been produced and is not required for subsequent processing of the executable. Furthermore, this process facilitates the commodification of computer software, as the easily-read intellectual property held within the source code is stripped at the compilation stage and the program logic is disguised by the sheer complexity of the resultant object code. The executable may then be sold as a licensed "product",[9] such as Microsoft Word, and users pay both to be able to use the software and to upgrade to new versions. Although it is possible to reverse-engineer object code back into source code, it is complex, time-consuming, and usually illegal.

Software that is supplied as an executable without source code, usually under restricted license terms, is known as proprietary software. The majority of available commercial software falls into this category (Stallman, 2003a). In contrast, software that is sold or distributed with the source code, which provides no restrictions on further copying and modification, is known as free software[10] or open source. "If it's not source, it's not software" (Stallman, 2003a). The FSF and OSM both believe that the original source code should always be made available. For different reasons they hold that the ability of an end user to be able to view the source code is important.[11] However, even though the OSM and FSF both use similar arguments for the benefits of freely available source code, they differ radically in their underlying philosophies.[12]

Software itself can be analysed using three analytical categories which this paper introduces:

1 *Use functions*: the ability and freedom to perform a specific or general computer-based task using the software.
2 *Prescriptive functions*: restrictions on what can be done with the software, usually by architecting into the software the delegation of explicit control and prescriptions onto the user (Feenberg, 1995).
3 *External functions*: actions outside the scope of the software; this latter category is used to distinguish between prescriptive functions and actions that cannot be performed because the software was not designed for the purpose, for example playing music on a word processor.

Proprietary software is sold on the value of its use functions. A word processor, for example, has the functionality to produce a wide variety of documents and letters. In contrast, software manufacturers often conceal the prescriptive functions built into the software. Due to pressure from content providers, copyright owners and patent holders, manufacturers are increasingly controlling usage of the software by restricting the actions of the user through the use of prescriptive functions.

A contemporary and highly contentious example of these techniques is digital rights management (DRM). DRM prevents users from carrying out unauthorized actions on copyrighted works often irrespective of the ownership or rights of the individual user (Lessig, 1999). Adobe Acrobat and E-books, for instance, have the ability to prevent the user from copying, changing or even quoting protected books. The software is delegated the legal restrictions of the copyrighted work and then prescribes these restrictions back on the user. The user is thus unable to perform activities that break the terms of the legal copyright.[13]

Additionally, prescriptive functions also raise privacy concerns, especially with the increased use of monitoring software, such as that increasingly installed in the workplace, which monitors for illicit actions by the user and reports them to the employer.[14] The ability to read the source and prevent the prescriptive potentials of software being implemented is one of the FSF's justifications for viewable source code.[15]

Coding and key signifiers

Both the OSM and the FSF use textual elements to articulate various aspects of discourse. They attempt to fix the polysemic elements which are shared between their discourses and create chains of equivalence (Phillips & Jørgensen, 2002). Additionally, they seek to place alternative articulations from each other within the field of discursivity and exclude them from the order of discourse (Fairclough, 1992; Phillips & Jørgensen, 2002). This temporary closure can never be definitive and fixed; nevertheless, the antagonistic discursive struggle can be dissolved through a hegemonic intervention (Phillips & Jørgensen, 2002). This will be explored further here.

By analysing text and interviews a preliminary coding will be outlined for 17 discourses from the two respective movements: a total of 52,088 words. The discourses

used concentrate on the texts of two key individuals: Richard Stallman, the founder of the FSF, and Eric Raymond, the founder of the Open Source Initiative.[16] These texts were chosen for their foundational value to the two movements' respective constituents and the individuals' importance as perceived founding fathers of the free software and open source movements (Bonaccorsi & Rossi, 2003; DiBona, Ockman, & Stone, 1999; Moody, 2002; Scoville, 1999). The texts are analysed using the text analysis markup system (TAMS), an open source discourse analysis software package (Weinstein, 2003). Key signifiers are drawn from the texts and they will be examined and compared together with their chains of equivalence in order to show how concepts concerned with identity, such as representation and group identity, and concepts concerned with conflict, such as antagonism and hegemony, are ordered discursively.

Discourses can also interpellate individuals by creating subject positions for people to occupy. They imply certain expectations about how to act, what to say and what not to say (Phillips & Jørgensen, 2002). Examinations of the discourses of the FSF and the OSM will demonstrate the subject positions within their discourses and how they are constructed. The rights and obligations of these positions are different within the two traditions and the hierarchical relationships and interaction will be outlined. These have social and political implications (Phillips & Jørgensen, 2002). For example, the FSF utilizes a discourse of ethics and a discourse of freedom (Stallman, 2003e), whereas the OSM draws on discourses of neoliberalism and technical efficiency (Raymond, 2001a).

This paper will examine the implications of their positions and attempt to point to their theoretical and philosophical origin. Further, the two movements are both utilising a model for the production of knowledge, both in terms of a support for the claims to 'true' knowledge and in terms of their understanding of the relationship to the external world. These philosophical positions imply an underlying conception of agency and epistemology (Linstone & Murray, 2002) and will be examined in turn.

Broadly speaking, this paper intends to examine two major strands in the discourse of the OSM and FSF. Firstly, they are based firmly within the community of technologists and are committed to the social good that open or free software can provide, but differ radically in their respective assumptions about how this good is to be achieved (Scoville, 1999).[17] Secondly, they reflect wider societal questions about technological determinism, efficiency and the democratization of technology. Each movement condenses these debates into strongly differing approaches to technological progress and the legitimacy of technocracy. As these issues are considered important contemporary questions (Feenberg, 2002; Lessig, 1999), this paper seeks to place these arguments within a broader framework and offer some conclusions and recommendations as to their wider application.

Free Software Foundation

The FSF uses discourses from enlightenment philosophy, communitarianism and the collegiate ideals of the academic and scientific communities (Bezroukov, 2003; Kelty, 2001) both intertextually and interdiscursively to present a strong moral position (Stallman, 1993, 2003d, 2003e). The FSF appears to take a deontological

position in regard to personal ethics and a Kantian flavour is readily seen in the calls to abide by the general moral laws of the FSF. These 'laws' can serve both as a guide to individual action (Stallman, 1993) and as a means of conflict resolution within the movement by making manifest a shared ethical outlook (Elliott & Scacchi, 2002).

A Kantian notion of a categorical imperative also seems to underlie the philosophical foundations of the FSF: What is ethical for the individual must be generalizable. In terms of coding, the ethic of sharing all code with others within the project is unambiguously Kantian in principle. Some writers have characterized Stallman as being driven by aesthetic rather than ethical reasoning (Harvey, 2003). However, this seems to be based on a misunderstanding of Stallman's position regarding efficiency and utility. "We in the free software movement recognize these practical benefits, and they are nice, but they are not the most important issue. More important are the ethical and political aspects" (Stallman, 2003d).

The FSF uses what I will call a discourse of ethics and a discourse of freedom. The discourse of ethics outlines a basic philosophical position whereby access to the underlying source code of a software object is a human right. As Stallman outlines: "I consider [free software] a human right, and thus a moral norm" (Personal communication, 28 June 2002). Additionally, Stallman believes that freedom is intrinsically linked to the FSF's aims and that freedom of the individual is freedom from the tyranny of technology (Stallman, 2003e).

The FSF presents their philosophy by hypertext referencing to other texts.[18] This seems to represent not only a legitimation of knowledge by reference to sources, but also an application of Stallman's ideals of personal efficiency (Williams, 2002) and the importance of crediting and acknowledging others (Kelty, 2001). This manifest interdiscursivity is demonstrated on the FSF web pages in the following beliefs:

1 that motivation is not a vulgar behaviourism (Kohn, 1987);
2 that technical efficiency is not the only driver of technology progress (Stallman, 2003d);
3 that self-interest is not the only motivator for hackers (Fueston, 1998).

Interestingly, in a Kantian vein, Stallman also argues that "the law should conform to ethics, not the other way around" (Stallman, 1992).

The FSF also appears to take a Kantian approach to the production of truth. This is a contributory theory of knowledge that encourages many different and competing judgements and solutions to be applied to the problem area (Linstone & Murray, 2002). This approach allows many 'informed' individuals from different disciplines and specialities to contribute information to the project and consequently allows a broader definition of the problem area and encourages a goal-oriented methodology. This is demonstrated both in the general nature of the FSF itself, which seeks to maximize freedom, and in its specific aim to produce a non-proprietary version of Unix that is completely unrelated to proprietary versions (Stallman, 1993).

The FSF actively calls for contributions and participation from interested parties without necessarily specifying a physical platform for implementation (Stallman, 1993). Indeed, the radically open modular nature of the design has the potential to encourage a democratization of technology due to the creation of different and

competing implementations. This is one of the hallmarks of the free software and open source movements and many websites serve both to fork projects and discuss different projects' relative pros and cons (Slashdot, 2003).

The preliminary key signifiers identified in this analysis within the FSF discourse are code, freedom, power, progress, community and rights.

Code

Code is a nodal point. In other words it is a privileged sign around which the other signs are organized. Other signs acquire their meaning from their position in relation to this nodal point.

> Programmers normally work with the 'source code' for a program, which is written in a programming language such as Fortran or C...It is designed to help programmers read and change programs...Source code is useful (at least potentially) to every user of a program. But most users are not allowed to have copies of the source code...It leads to resignation and discouragement, which can spread to affect other aspects of one's life.
>
> (Stallman, 1992)

> [W]ho should control the code you use – you, or an elite few? We believe you are entitled to control the software you use, and giving you that control is the goal of Free Software.
>
> (Kuhn & Stallman, 2001b)

For the FSF, code is a public good that is a social constructed phenomenon and should be freely shared. Code is more strongly associated with the social practice of coding, in other words the production of code in a social network.

> Suppose that both you and your neighbor would find it useful to run a certain program. In ethical concern for your neighbor, you should feel that proper handling of the situation will enable both of you to use it. A proposal to permit only one of you to use the program, while restraining the other, is divisive; neither you nor your neighbor should find it acceptable...This is psychosocial harm associated with the material harm of discouraging use of the program.
>
> (Stallman, 1992)

Conclusions are drawn about the ethics of coding, sharing, contributing and the importance of the publicness of ideas. As Stallman is concentrating on the social practice, it seems logical that this cannot be protected or withheld from the group that was instrumental in forming the ideas in the first place.

> In any intellectual field, one can reach greater heights by standing on the shoulders of others. But that is no longer generally allowed in the software field – you can only stand on the shoulders of the other people in your own company.
>
> (Stallman, 1992)

Copyright is therefore problematic as it restricts other's ability to use information and further both their own and society's progress. "[T]he power to restrict changing or copying it – is obstructive. Its negative effects are widespread and important. It follows that society shouldn't have owners for programs" (Stallman, 1992).

Stallman also explicitly attacks the notion of patents due to the monopoly they give in the realm of ideas (DiBona et al., 1999; Garfinkel, Kapor, & Stallman, 1991; Stallman, 1991a).

Rights

"The concept of rights and particularly a liberal notion of human rights underlie the constructions of access to source code as a basic human right" (Stallman, 2002b).

Stallman utilizes a strong concept of rights drawn from the American constitution to justify this position. "[T]he idea of inalienable rights embodied in the GNU GPL comes from the founders of the United States" (Stallman, 2002b). Further: "The ethical response to this situation is to proclaim freedom for each user, just as the Bill of Rights was supposed to exercise government power by guaranteeing each citizen's freedoms" (Kuhn & Stallman, 2001b).

However, Stallman is careful to delimit the potential for a conception of natural rights for property ownership from a Lockean tradition.

> The idea of natural rights of authors was proposed and decisively rejected when the US Constitution was drawn up. That's why the Constitution only permits a system of copyright and does not require one; that's why it says that copyright must be temporary.
>
> (Stallman, 2003f)

For instance he states: "The real established tradition of our society is that copyright cuts into the natural rights of the public – and that this can only be justified for the public's sake" (Stallman, 2003f).

This is an important distinction for Stallman who is keen to emphasize the corrosive effect of copyright and patents on freedom of expression. By this means, the argument for a stronger moral claim for the freedom of access to the source code is constructed. This forms part of the justification for free software by linking to the wider discourse of freedom that Stallman uses within the texts.

Community

Within the discourse of ethics, Stallman identifies that being an active member of a civic community and the act of sharing with a neighbour is highly important.

> We look at what permits a good way of life, and at how useful programs can foster a community of goodwill, cooperation, and collaboration. Our criteria for free software specify the freedoms that a program must offer its users so that they can cooperate in a community.
>
> (Kuhn & Stallman, 2001b)

For the FSF, "non-free software is a social problem and free software is the solution" (Stallman, 2003e). The discourses tend to value a "good life" that promotes positive values. For example: "Our criteria for Free Software specify the freedoms that a program's users need so that they can cooperate in a community" (Kuhn & Stallman, 2001b).

The conception of "social good" for free software includes the importance of the social and the communicative experience of coding. This social sharing manifested within the free software movement is built on trust and reliance on others to provide improvements and ideas freely to the project. "The freedom to improve the program, and release your improvements to the public, so that the whole community benefits (freedom 3). Access to the source code is a precondition for this" (Kuhn & Stallman, 2001b).

This conception of the social good is strongly communitarian and privileges both a vision of a social order that assigns social rights and responsibilities, and one that is fair and equitable. Each contributes code to the project according to their ability and takes code according to their need: "above all society needs to encourage the spirit of voluntary cooperation in its citizens" (Stallman, 2003f).

The spirit of voluntary cooperation, which Stallman believes is part of the desire to help your neighbour and engage in civic spirit, should be promoted and encouraged. "Programmers also suffer psychosocial harm knowing that many users will not be allowed to use their work. This leads to an attitude of cynicism or denial" (Stallman, 1992).

Freedom

Within the discourse of freedom, Stallman outlines a number of reasons why software should be free, drawing on principles of natural rights and social rights. This is one of the principle differences between the open source and free software movements. "'Free software' is a matter of liberty, not price. To understand the concept, you should think of 'free' as in 'free speech', not as in 'free beer'" (Stallman, 2003b).

Fueston (1998) states: "The presentation of the ideals of freedom and sharing imply an ethical choice for the software developer to take to his fellow developers".

> That's true: talking about freedom, about ethical issues, about responsibilities as well as convenience, is asking people to think about things they might rather ignore. This can trigger discomfort, and some people may reject the idea for that. It does not follow that society would be better off if we stop talking about these things.
>
> (Stallman, 2003e)

Stallman (2203e) goes on: "If you feel that freedom and community are important for their own sake – not just for the convenience they bring – please join us in using the term 'free software.'" The freedom that Stallman envisages is formed around the idea of being able to shape and change both one's own destiny and also the tools that are used along the way. This strong conception of 'free' includes not just ideals of freedom of speech and freedom of assembly but also the dangers to freedom of

thought if ideas themselves have restrictions on their usage. "To stop using the word 'free' now would be a mistake; we need more, not less, talk about freedom" (Stallman, 2003e).

Stallman identifies a common engineering problem, namely the danger of being unable to question 'black boxes' that cannot be opened to check their contents. This is very close to the concept of a technical code (Feenberg, 2002): a moral obligation imposed on humans by the delegation to machines.

> What does society need? It needs information that is truly available to its citizens – for example, programs that people can read, fix, adapt, and improve, not just operate. But what software owners typically deliver is a black box that we can't study or change.
>
> (Stallman, 2003f)

Quite simply, black boxes not only present the possibility of surreptitious spying or monitoring, but they cannot be repaired or changed and, most importantly for Stallman, improved, so that progress is hindered.

> I am working to build a system where people are free to decide their own actions; in particular, free to help their neighbors, and free to alter and improve the tools which they use in their daily lives. A system based on voluntary cooperation and on decentralization.
>
> (Stallman, 1992)

Power

The concept of power is associated with a Weberian concept of power over the user. Where one can use a proprietary software license to restrict the activities of the user or programmer, there is an act of power.

> Proprietary software is an exercise of power. Copyright law today grants software developers that power, so they and only they choose the rules to impose on everyone else – a relatively few people make the basic software decisions for everyone, typically by denying their freedom.
>
> (Kuhn & Stallman, 2001a)

Power to control draws particularly on the notion of prescription and the inability to check the existence of these prescriptive functions within the source code. "Current copyright law places us in the position of power over users of our code, whether we like it or not" (Kuhn & Stallman, 2001b).

Via black boxes, the user is forced to act under the control of the proprietary software manufacturers with no recourse to appeal. This is a fundamentally undemocratic moment, and forms part of the argument for the transparency of code as a desirable and democratic approach to social life. "I shouldn't have the power to tell you not to do these things. No one should" (Stallman, 2003f). "In this conception of power arguments are usually focused on ways to reduce the monopoly

on the mechanisms supplying this power, for example preventing copyright or patents" (Kuhn & Stallman, 2001b). "The ability to get inside the technical device means that the code can be changed and controlled and that the freedom of the individual to choose is paramount" (Stallman, 1992).

> The idea of the sovereignty of the individual naturally questions the implementation of prescriptive architectural properties within computer code. Restrictions on the natural freedoms of the user are a restriction on their freedom of choice. The FSF is therefore strongly opposed to the use of prescriptive functions in software.
>
> (Stallman, 2003d)

Progress

The FSF uses the signifier "progress" to indicate that without the collective provision of free software an enlightenment ideal of progress would be lost. Indeed, Stallman approvingly quotes the US Constitution, stating that copyright is designed to "promote the progress of science and the useful arts" (Stallman, 1992). Congress agreed that the rights of the public were temporarily suspended by the constitution of the public's rights through a system of copyright to further progress. Stallman explains: "It also states that the purpose of copyright is to promote progress – not to reward authors" (Stallman, 1992). There are no authorial rights in the constitution, merely temporary copyrights. For the FSF these rights were predicated on a system of property which by its very nature was limited, material and not easily copyable. These are contrasted with the virtual goods of the Internet which are unlimited, non-material and easily and freely copied.

> Our ideas and intuitions about property for material objects are about whether it is right to take an object away from someone else. They don't directly apply to making a copy of something. But the owners ask us to apply them anyway.
>
> (Stallman, 2003f)

Subject positions

Readily using the terms "we" and "they", the FSF utilizes the concept of in-group and out-groups to identify friends and enemies. These subject positions are treated as a dichotomy and the reader is assumed to be supportive of the FSF objectives, a friend and colleague, or if not an enemy. Most prominent is the attempt to set up a distinction between the free software and open source movements, when in fact they share many of the same authors and coders.

> We are not against the Open Source movement, but we don't want to be lumped in with them. We acknowledge that they have contributed to our community, but we created this community, and we want people to know this. We want people to associate our achievements with our values and our philosophy, not

> with theirs. We want to be heard, not obscured behind a group with different views. To prevent people from thinking we are part of them, we take pains to avoid using the word 'open' to describe free software, or its contrary, 'closed,' in talking about non-free software.
>
> (Stallman, 2003e)

The strong first person modality of the text and the use of the collective "we" implies an attempt to seek closure within the order of discourse and thereby excludes alternative or conflicting definitions or interpretations. The "other" that one might have identified oneself with is therefore excluded and the potential for conflict and overdetermination of the subject is avoided (Phillips & Jørgensen, 2002). Additionally, the text seeks to speak both for and to the group and in the process defines key signifiers such as code and freedom. Consequently, the FSF attempts to divide the social space of computer programmers, technologists and coders into groups along lines that further the FSF's aims and objectives and to fill the different master signifiers with their content. This results in a hegemonic struggle with the OSM for the contestation of the key terms and signifiers that are shared between the movements.

Open source movement

The OSM uses what I will identify as a discourse of technical efficiency and a discourse of neoliberalism. These are used intertextually and interdiscursively within the discourses of the OSM to legitimize and position their arguments as rational, natural and commonsense. Eric S. Raymond is one the founders of the OSM and he clearly differentiates the OSM position from that of the FSF, stating that "Open Source is not particularly a moral or a legal issue. It's an engineering issue. I advocate Open Source, because very pragmatically, I think it leads to better engineering results and better economic results" (Raymond, 2003b).

The OSM differentiate sharply between the technical, rational and objective sphere of software development and that of the political sphere (O'Reilly, 2002). The problem of freedom is seen as one of freedom to choose within a system of market relations (O'Reilly, 2001). In other words, the developer has the right to choose the licensing model and the user the right to choose to use the software. This confuses a consumerist notion of "economic freedom within a marketplace" (O'Reilly, 2001) with that of the more politicized notion of the essential human right to freedom of choice within all spheres (Kuhn & Stallman, 2001b).

Raymond outlines a crucial philosophical method called the 'Delphi effect' to justify the OSM's methodology (Raymond, 2001a). "Delphi may be characterized as a method for structuring a group communication process so that the process is effective in allowing a group of individuals, as a whole, to deal with a complex problem" (Linstone & Murray, 2002, p. 3)

First pioneered by Helmer and Rescher at Rand, the Delphi technique is an example of Lockean inquiry (Linstone & Murray, 2002). The key aspect of this theory is that truth is experiential. In other words, the truth content is associated entirely with its empirical content. Every complex proposition is reduced to simple observations and the validity of these is ensured by the freely obtained

agreement between different observers. This is an experimental, consensual system. This consensual system is best suited to an already informed and specialized knowledge community who share a 'core' body of knowledge. This characterizes the members of the OSM.

To build the system Raymond starts from a set of elementary empirical judgments, raw data, observations or sensations. From these is built a network of expanding, increasingly more general networks of factual propositions. The final system is subjected to agreement by a group of experts. The raw data is granted a prior existential status. This is demonstrated explicitly in his paper *Homesteading the Noosphere* (Raymond, 2003a) which argues for a property-based system on the Internet.

The OSM uses the key signifiers code, the market, freedom, efficiency, property and the individual within their discourses which are discussed below.

Code

For the OSM, code is also a key nodal point around which the other signs are organized. This sign refers to an empirical object, the source code, which, for Raymond, exhibits clear property rights. However, for code to be allocated property rights Raymond attempts to identity a 'homesteading' of a realm of ideas that is undertaken by the programmer.

> The 'noosphere' of this paper's title is the territory of ideas, the space of all possible 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. Hence 'homesteading the noosphere', which is what every founder of a new open-source project does.
>
> (Raymond, 2003a)

Raymond constructs a rational choice model to explain the desire to produce code and this conception of property runs into difficulty taking into account the infinitely copyable nature of code. This seems to undermine the scarcity requirement for the normal functioning of a property related market, and so he seeks to change the property value of code from the actual "copy" to the ownership of the "project" and therefore its history, direction and future (Raymond, 2003a).

Indeed, this enables an explanation of the single open source project and also allows a discussion of the taboos of "forking", or appropriating a project without the permission of the previous project owner. As he explains: "If use were the only issue, there would be no taboo against forking, and open-source ownership would not resemble land tenure at all" (Raymond, 2003a).

For Raymond, code is hence a slightly more complex concept taking in not only the source code itself but also the structure, control and direction of the entire open source project. The construction of a web page thus begins to represent the marking of territory for Raymond as it is a location on the Internet where the project is managed and users congregate to seek copies and information (Raymond, 2003a).

Efficiency

Using a discourse of technical efficiency, Raymond identifies the following key ideas:

1 Technical efficiency is derived from many people working simultaneously on a project – "Many eyeballs make bugs shallow" (Raymond, 2001a)
2 The technocratic belief that the best technical solution is the most efficient (Raymond, 2001a)
3 The inefficiency of centralized control systems and big social projects (Raymond, 2001a)
4 The fact that the market is a superior mechanism for delivering goods and services (Raymond, 1999a)

> While a minority of hackers does indeed remain hostile to the profit motive, the general willingness of the community to cooperate with for-profit Linux packagers like Red Hat, SUSE, and Caldera demonstrates that most hackers will happily work with the corporate world when it serves their ends.
>
> (Raymond, 1999a)

For Raymond, the profit motive is the greatest source of technical efficiency and this explains his desire to construct a property system on the Internet which could prevent the so-called tragedy of the commons (Raymond, 1999a, 2003a).

Freedom

Using a discourse of neoliberalism, Raymond identifies several important concepts related to his ideas of freedom. He appears to use a form of ethical egoism and ideas drawn from psychological egoism as a justification for a normative stance stating that because we are *actually* acting in selfish ways, we therefore *should* act selfishly.

For Raymond, open source software projects are started because of the needs of an individual, whether fixing a bug or to perform a function. "Every good work of software starts by scratching a developer's personal itch" (Raymond, 2001a).

He dismisses the idea that people may wish to write software altruistically for others. For Raymond, the truth is that altruism does not exist – "One may call their motivation 'altruistic', but this ignores the fact that altruism is itself a form of ego satisfaction for the altruist" (Raymond, 2001a).

For Raymond, the most important type of freedom is economic freedom. Consequently, rational choice and economics are used as explanatory devices to explain how the uncoordinated action of many programmers working on a project mirrors that of the "invisible hand" of the market (Raymond, 1999a). In fact, Raymond explains that the term "bazaar" is synonymous with the market (Raymond, 2001a). Hence the importance of the privatization of the source code and the fact that every project has an owner (Raymond, 1999a, 2001a, 2003a).

As the OSM regards freedom entirely within the realm of an individual's economic freedom and their freedom to work on projects, Raymond does not question the possibility or dangers of prescriptive functions. For example, the open source definition states that "the license shall not restrict any party from selling or giving

away the software as a component of an aggregate software distribution containing programs from several different sources" (Perens, 2003). This is in marked contrast to the FSF's absolutist prohibition on the mixing of different forms of software and the resulting viral nature of the GNU general public license (Stallman, 1991b, 2002a).

Property

The intertextual and interdiscursive elements within the discourse of the OSM borrow heavily from Lockian philosophy, Adam Smith, Ayn Rand and other libertarian and rational choice theorists like Mancur Olson. Raymond has a strong notion of utilitarianism and it appears that rational choice theory forms a basis of his world view (Raymond, 1999a, 2001a, 2003a).

However, a strongly evolutionist thread runs through Raymond's discourses that seeks to give deterministic causes. For example:

> It is sometimes fashionable to describe human property as an arbitrary social convention, but this is dead wrong. Anybody who has ever owned a dog who barked when strangers came near its owner's property has experienced the essential continuity between animal territoriality and human property.
>
> (Raymond 2003a)

Additionally, Raymond has stated neoliberal beliefs in the rejection of altruism and seeks to find cooperative behaviour as an accidental by-product of the interactions of free agents in a competitive market. Clearly this ignores many of the troublesome theoretical problems of the inability of rational individuals to actually work together at all due to the so-called free-rider problem (Barnes, 1995; Olson, 1971). Drawing approvingly on the work of Locke and Adam Smith he explains the success of the OSM using an analogy to the invisible hand of the market, thus seeking to naturalize the process. "We have examined the customs which regulate the ownership and control of open-source software. We have seen how they imply an underlying theory of property rights homologous to the Lockean theory of land tenure" (Raymond, 2003a).

For Raymond, open source addresses the problems of large-scale computer programming problems and provides a technically efficient means to bypass the problems and inefficiencies of highly centralized bureaucratic structures (Raymond, 2001a).

The market

The conception of social good with regard to open source software includes the importance of the technical advantages achieved in terms both of software quality and efficiency of the approach and the provision of public goods through the invisible hand of the market (Raymond, 1999a). This social sharing manifested within the OSM is built on trust and the reliance on others to provide improvements and ideas selfishly but freely into the project (Raymond, 2001a).

The individual

The conception of the social good is strongly neoliberal and libertarian. It privileges both a vision of a highly individualistic social order, strongly influenced by ideas from Darwinist and evolutionary thinking, and believes that collective goods can be produced through the selfish action of individuals. "[T]he closed-source world cannot win an evolutionary arms race with open-source communities that can put orders of magnitude more skilled time into a problem" (Raymond, 2001a).

The individual owner is a key figure in the OSM (Raymond, 2001a). Constructed as an almost absolutist monarchy it is clear that democracy is far from the mind of the hierarchical structure that Raymond identifies as being most effective at providing leadership (Raymond, 2003a). "The owner makes all decisions and collects all credit and blame. The only possible conflicts are over succession issues – who gets to be the new owner if the old one disappears or loses interest" (Raymond, 2003a).

Subject position

Using nominalization, passivization and objective modalities, the OSM's discourse presents a rational, objective and non-affective discourse that seeks to appeal to the technologically and scientifically trained developer community.[19] Presenting opinions as facts through the removal of subjective modality allows the OSM to outline and draw on a discourse of natural law. The laws themselves are constructed by the systemic mystification of agency. For example, markets are presented as subject to laws of nature, Lockean individuals precede society and the evolutionist law of the jungle is not only an explanatory device but presented as a palliative to society's economic ills.

Interestingly, Raymond avoids divisive terms such as 'us' and 'them'; there is no group collectives that he claims to be a spokesperson for. Instead, he privileges the individual, and by the use of 'scientific' reasoning and pseudo-anthropological methods attempts to uncover the truth about hackers and coders in their open source endeavours.

Comparative analysis

Although not all key master signifiers are shared between the discourses in the FSF and the OSM, there is a struggle at the level of discourse and each movement is self-conscious of the other as they present their explanations and interpretations (Kuhn & Stallman, 2001b; O'Reilly, 2001, 2003; Raymond, 2001b, 2001c; Stallman, 2003e). In particular, contestation of the concepts of the individual, property's status as being either collective or private, and the best way of maximizing freedom and prosperity, seem to be of particular concern (Scoville, 1999).

For the FSF, code is constructed as a public or collective good that is akin to a utility or law (see Lessig 2002 for a development of this idea). The development of GNU/Linux is the key working example, a collective project that has been shared and worked on freely and remains firmly with the copyleft principles of the FSF. The collective system of GNU/Linux support and development represents the

exemplar of this approach. Some individuals donate tools, for example Stallman created huge amounts of GNU tools and libraries and others supplied key operating system modules, for example the LINUX kernel or the desktop interface. The Unix system is a highly modular design that allows this kind of collective effort to be easily organized by combining multiple software sources.

In the OSM, code is property owned by an individual who has the right to control and develop it, but this is strictly associated with the notion of the project itself rather than just the underlying source code. Linus Torvalds, the creator of Linux, is the exemplar of the vision of Raymond and the OSM. Within the OSM literature, Linus is the epitome of the individual programmer creating from scratch a Unix system, LINUX. He supplied the skills, the vision and he remains a key figure in directing and managing the project – often described as a "benevolent dictator".[20]

In terms of freedom, it can be noted that the freedom to use, modify, read and copy software, designated as a collective good, is of key concern to the FSF. "Free as in Freedom, not Beer" (Stallman, 2003e) is the slogan that the FSF has made famous. The OSM, however, is concerned with the freedom of the individual to work on a project that is of particular selfish interest: "an itch that needs to be scratched". If this is useful to others, then that is the free-market meeting needs, not a collective, centrally planned project. The key individual, the lead developer, is of the utmost importance here, a strong Randian character that pulls everybody else along by sheer force of will and power (Rand, 1992). Comparisons between this stereotype of the OSM developer, or benevolent dictator, and the characters within *Atlas Shrugged* (Rand, 1992) do not appear to be accidental.

In terms of progress, both movements have a strong modernist technocratic model of linear progress. However, they differ in their conception of the ends of the project. The FSF appears to have an Enlightenment ideal of progress as a light to shine on the darkness of tradition, a collective good for all humanity. Comparisons have been made between the FSF philosophy and the principles of academic and scientific research publishing, both strongly influenced by Enlightenment philosophy. In contrast, the OSM has a more brutal ideal of capitalist progress and technical efficiency – to achieve a more efficient and profitable solution without regard to values is optimal.

It is interesting to note that some early open source founders have returned to the FSF due to their discomfort with the direction of the OSM (Perens, 1999). Indeed, Raymond's eccentric and often strong libertarian positions on issues from gun control to terrorism have alienated many potential supporters (Raymond, 1999b, 2002a, 2002b). The thread of libertarianism runs deeply through all of his writings and it is clear that this has informed his disguised attack on Stallman in *The Cathedral and the Bazaar* (Raymond, 2001a). Raymond's belief in the power of rational choice, namely uncoordinated selfish action to produce collective goods, utilizes an American anti-government, anti-centralist rhetoric. His assumptions about minimal government – sometimes he actively calls for no government at all (Raymond, 1999b) – borrow ideas about the sovereignty of the individual based on Locke's ideas of pre-governmental life (Raymond, 2003a). For someone keen to avoid values and ethics and concentrate on the technical and rational, his ideas are permeated with his particular ideological position.

Conclusions

This paper has examined the discursive struggle taking place between the OSM and the FSF. Using discourse analysis it has demonstrated that there is an attempt to achieve hegemony by fixing the elements within the discourses surrounding the production of code. This could have wider ideological implications for the Internet community and, indeed, society at large.

Through an analysis of the discourse produced by these two movements, it appears that the OSM is providing a more convincing order of discourse to the technologist community (Scoville, 1999). This is due to subtle use of intertextuality that uses wider arguments from neoliberal economics and technocratic discourses and the more overt interdiscursivity from wider neoliberal texts (Raymond, 2003a). This is opposed to the FSF's "appealingly utopian – and perhaps quixotic – notion that all information should be shared" (Scoville, 1999).

Indeed the importance of hegemony within this order of discourse has greater political and philosophical implications when considered within the context of the growth in popularity of ideas surrounding anti-copyright, copyleft,[21] the public domain and issues of freedom and democratization. This is demonstrated in the growth of free and open source projects ranging from hardware designs to record labels, walkmans, books and online discussion sites.[22] It has even been considered an issue of national security as demonstrated by the debates regarding software within the governments of Brazil, India, China and South Africa to avoid dependence on western, mainly American, proprietary computer software products, particularly in governmental, educational and military usage (Berry & Moss, 2004; Weber, 2003).

Many engineers attempt to take a consequentialist position in regard to technology. They are trained in rational positivistic approaches to solving problems and concentrate on mean-end rationality (Feenberg, 2002).[23] This instrumental approach informs their training and influences their personal standpoint (Barbrook & Cameron, 1995; Slashdot, 2001a, 2001b). For them, the OSM offers a commonsense approach both in its language and philosophy, both by seeking "obvious" provable solutions and explicitly positioning itself apart from political debate: "OSS is not about politics, it's about software. Don't lose sight of that" (Slashdot, 2001a). The OSM and its followers generally view politics within the technical sphere with misgivings (O'Reilly, 2002). By positioning technologists' best practice as apolitical they seek to create an opposition to the FSF's value based discourses even if they agree with its other technical discursive elements.

The FSF, with its emphasis on the subject positions 'us' and 'them' requires the reader to take a deontological ethical position. To many of its readers this approach seems old-fashioned and unscientific (Scoville, 1999). Without a convincing and wider call to politicize the technical sphere more generally, the FSF will continue to be seen as "wildly utopian" (Scoville, 1999). In order to contest this hegemonic discourse, the FSF will need to undertake a more concerted attempt to deal with the strongly objective modality of the discourse of the OSM and the implication that technology is a purely value-free activity.

The FSF could address the deeply divisive nature of its discourse by positioning itself as having both a moral and ethical position, and a scientific and technical one.

However, this would be to concede the possibility of a non-political technical activity and a value-free technical discipline and risk undermining its moral and ethical position.

Indeed, it is the view of this paper that the FSF should seek to widen its discourse from deontological ethics and community shared processes for the production of social goods to a wider discourse of democracy. This could take the form of a concentration on particular democratic freedoms that the free/libre software movements offer in terms of participation, value sensitive design and transparency (Nissenbaum & Howe, 2003). It could also draw on discursive elements from wider democratic debates in society (Barber, 1984; Habermas, 1988, 1992, 1997), theories about the democratization of technology (Feenberg, 2002; Sclove, 1995), issues surrounding public debate for steering technological policy making (Margetts, 1996; Winner, 1986) and the need for public involvement in the production of far-reaching and highly invasive technologies such as is the case in genetically modified (GM) foods and the environmental movement (Feenberg, 1995; Winner, 1986).

As technology increasingly colonizes and structures more aspects of our lives it is becoming increasingly important that the constitutive nature of technology as socially shaped is recognized (Kesan & Shah, 2002; MacKenzie & Wajcman, 1999). If computer code is analogous to law (Kesan & Shah, 2002; Lessig, 1999), then it is clear that without some form of democratic accountability the code-based regulation of human behaviour will continue to lack legitimacy (Habermas, 1988; Sclove, 1995). It is therefore the view of this paper that within the discursive struggle between the FSF and the OSM is an important challenge for wider society to recognize that values are being instantiated within technological forms that can and should be contested before they become sedimented. Introducing democratic accountability to code may well be the democratic challenge of the twenty-first century and steering the implementation of technological artefacts will increasingly contribute to our ability to keep our future open and democratic.

Acknowledgements

I would like to thank Richard Stallman, Andrew Feenberg, Tiziana Terranova, Richard Barbrook, Graham Seaman, Caroline Bassett, John Drury, Eric Raymond, Lee Salter and Giles Moss for their comments and criticisms during the writing of this paper.

Notes

1 This paper uses the term "order of discourse" to represent a specific area of discourse within the field of discursivity following Fairclough (1995) and Phillips and Jørgensen (2002). Within the paper, the order of discourse is the use, production and ethics surrounding computer code.

2 Computer programming code is discussed in detail in the section entitled 'Source code', but generally speaking this paper is concerned with the textual artefacts of programming, including source code, documentation, programming models, formal languages and related discursive products.

3 Whilst stressing a critical discourse analysis perspective to develop analysis within a coherent discursive moment, this paper views critical discourse analysis as a constellation of approaches and methods to examine hermeneutically the radical incompleteness and contingency of hegemonic discourse (Laclau, 2001; Laclau & Mouffe, 2001; Phillips & Jørgensen, 2002).

4 For an example of what machine code looks like see Stallman (2002a, p. 4)

5 The term "hacker", in contrast to popular media portrayals of illegal and illicit usage of computers, actually has traditionally had more positive connotations within computer programming, referring to expert programmers (Himanen, 2001).

6 "Program" is the term for a totality that is a functional unit of programming code that can be executed. Programs are often made up of sub-programs called functions, procedures and methods; these can be linked together to create higher level abstractions. A program is usually represented in source code and compiled into an executable.

7 The opposite of this is a method of program disguising known as "obfuscation". This is sometimes used in security conscious programming environments, whereby comments are removed so they do not assist comprehension of the code logic. Additionally, complex and confusing variable, function and method names are chosen and strange and non-standard formatting are all utilized to confuse the reader.

8 A compiler is a software tool that transforms source code into an executable.

9 Software is sold on the basis of a license to use it. However this does not mean that the buyer owns the software. In fact the Microsoft license explicitly states: "The OS components are licensed, not sold" (Microsoft, 2003).

10 Free software is also known as free/libre software.

11 For a detailed history of the free software and open source movements, see Lerner and Tirole (2002) and Moody (2002)

12 Although free software and open source have both been criticized for being "communist", neither is explicitly anti-capitalist or anti-commercial in approach to source code, perhaps reflecting their North American biases. Interesting developments regarding anti-capitalism and Marxist approaches to free software are being explored elsewhere although outside the scope of this paper (Richardson, 2003).

13 This has led to an increasing interest in the architectural similarity of code to law (Lessig, 1999).

14 This is increasingly the case with employee monitoring software that records the computer activities of staff using the web or sending email. For examples see Himanen (2001) and Spectorsoft (2003).

15 As some open source licenses allow the mixing of open and proprietary software, prescriptive functions cannot necessarily be prevented in an open source release.

16 The importance of these two individual's opinions on their respective groups is itself an interesting avenue for future research although outside the scope of this paper.

17 Both groups share a strong conception of linear progress and modernity (Raymond, 2001a; Stallman, 1993). However the extent to which communicative concerns can override the technical code (Feenberg, 1995) is examined further in the section entitled 'Coding and key signifiers'.

18 See http://www.gnu.org and http://www.stallman.org for examples of this hypertext approach to interdiscursivity.

19 For an examination of the extent to which persuasion plays a role in the building of technical systems see Bijker, Hughes, and Pinch (2001) and MacKenzie and Wajcman (1999).
20 An example of this is the benevolent dictator for life (BDFL) on the Python project (Python, 2003).
21 Copyleft provides certain freedoms to a user, stating that when redistributing an object of software, you cannot add restrictions to deny other people the ability to copy, use and modify it (Stallman, 2003b).
22 See for example Creative Commons (2003), *The Libre Society Manifesto* (Berry & Moss, 2003), LOCA records (Atton, 2003; LOCA-Records, 2003) and Stallman (2003c).
23 The social shaping of technology and social construction of technology literature can be a useful corrective to technologists' own idealized justifications of how technology is developed (Bijker et al., 2001; MacKenzie & Wajcman, 1999).

References

Atton, C. (2003, November 4). Talking about a quiet revolution. *The Financial Times*, p. 8.

Barber, B. R. (1984). *Strong democracy: Participatory politics for a new age*. California: University of California.

Barbrook, R., & Cameron, A. (1995). *The Californian ideology*. Retrieved June 21 2003, from http://www.alamut.com/subj/ideologies/pessimism/califIdeo_I.html.

Barnes, B. (1995). *The elements of social theory*. London: UCL Press.

Berry, D. M. (2003). *Internet research: Privacy, ethics and alienation – A free/libre and open source approach*. Retrieved January 21 2003, from http://www.sussex.ac.uk/Users/hbp17/.

Berry, D. M. (2004). *The political economy of open source: Immaterial labour and informational capitalism*. Retrieved March 15 2004, from http://www.sussex.ac.uk/Users/hbp17/.

Berry, D. M., & Moss, G. (2003). *The libre manifesto: Towards a libre society*. Retrieved June 21 2003, from http://www.libresociety.org.

Berry, D. M., & Moss, G. (2004). *Democratising digital government: The democratic implications of adopting open source software in government*. Retrieved January 10 2004, from http://www.sussex.ac.uk/Users/hbp17/.

Bezroukov, N. (2003). *Open source software development as a special type of academic research (critique of vulgar Raymondism)*. Retrieved June 16 2003, from http://www.firstmonday.dk/issues/issue4_10/bezroukov/index.html.

Bijker, W. E., Hughes, T. P., & Pinch, T. (2001). *The social construction of technological systems*. Cambridge: MIT Press.

Bonaccorsi, A., & Rossi, C. (2003). *Comparing motivations of individual programmers and firms to take part in the Open Source movement. From community to business*. Retrieved December 14 2003, from http://opensource.mit.edu/papers/bnaccorsirossimotivationlong.pdf.

CreativeCommons. (2003). *Creative commons*. Retrieved February 1 2003, from http://www.creativecommons.org.

Di Bona, C. Ockman, S., & Stone, M. (Eds.) (1999). *Open sources: Voices of the open source revolution*. Cambridge: O'Reilly.

Elliott, M. S., & Scacchi, W. (2002). *Communicating and mitigating conflict in open source software development projects*. Retrieved June 22 2003, from http://www.ics.uci.edu/~wscacchi/

Fairclough, N. (1992). *Discourse and social change*. Cambridge: Polity.

Fairclough, N. (1995). *Critical discourse analysis: The critical study of language*. London: Longman.

Feenberg, A. (1995). *Alternative modernity: The technical turn in philosophy and social theory*. London: University of California Press.

Feenberg, A. (2002). *Transforming technology: A critical theory revisited*. Oxford: Oxford University Press.

Fueston, L. (1998). *Self-interest*. Retrieved June 25 2003, from http://www.fsf.org/philosophy/self-interest.html

Garfinkel, S. L., Kapor, M., & Stallman, R. (1991). *Why patents are bad for software*. Retrieved June 25 2003, from http://lpf.ai.mit.edu/Links/prep.ai.mit.edu/issues.article

Habermas, J. (1988). *Legitimation crisis*. London: Polity.

Habermas, J. (1992). *The structural transformation of the public sphere* (T. McCarthy, Trans.). Cambridge: Polity.

Habermas, J. (1997). *Between facts and norms* (T. McCarthy, Trans.). Cambridge: Polity.

Harvey, B. (2003). *What is a hacker?* Retrieved June 25 2003, from http://www.cs.berkeley.edu/~bh/hacker.html

Himanen, P. (2001). *The hacker ethic and the spirit of the information age*. London: Vintage.

Kelty, C. M. (2001). *Free software/free science*. Retrieved June 21 2003, from http://www.firstmonday.dk/issues/issue6_12/kelty/

Kesan, J. P., & Shah, R. C. (2002). *Shaping code*. Retrieved December 10 2003, from http://opensource.mit.edu/papers/shah.pdf

Kohn, A. (1987). *Studies find reward often no motivator: Creativity and intrinsic interest diminish if task is done for gain*. Retrieved June 25 2003, from http://www.fsf.org/philosophy/motivation.html

Kuhn, B. M., & Stallman, R. (2001a). *Freedom or power*. Retrieved June 25 2003, from http://www.gnu.org/philosophy/freedom-or-power.html

Kuhn, B. M., & Stallman, R. (2001b). *Freedom or power?* Retrieved June 26 2003, from http://linux.oreillynet.com/pub/a/linux/2001/08/15/free_software.html

Laclau, E. (2001). *The philosophical roots of discourse theory*. Retrieved September 10 2003, from http://www.diskurs.dk/skaermkursus/skaerm4eng.pdf

Laclau, E., & Mouffe, C. (2001). *Hegemony and socialist strategy: Towards a radical democratic politics*. London: Verso.

Lerner, J., & Tirole, J. (2002). Some simple economics of open source. *The Journal of Industrial Economics*, *L*(2).

Lessig, L. (1999). *Code and other laws of cyberspace*. New York: Basic Books.

Lessig, L. (2002). *The future of ideas: The fate of the commons in a connected world* (2nd ed.). New York: Vintage.

Linstone, H. A., & Murray, T. (2002). *The Delphi method: Techniques and applications*. New York: NJIT.

LOCA-Records. (2003). *LOCA records mission*. Retrieved January 14 2003, from http://www.locarecords.com

MacKenzie, D., & Wajcman, J. (1999). *The social shaping of technology*. Buckingham: Open University Press.

Margetts, H. (1996). The implications for democracy of computerisation in government. In P. Hirst & S. Khilnani (Eds.), *Reinventing democracy* (pp. 70–84). London: Blackwell.

Microsoft. (2003). *Supplemental end user license agreement — Windows system resource manager for Microsoft Windows Server 2003, Enterprise edition and Datacenter edition*. Retrieved December 14 2003, from http://www.microsoft.com/windowsserver2003/downloads/wsrmeula.mspx

Moglen, E. (1999). *Anarchism triumphant: Free software and the death of copyright*. Retrieved June 20 2003, from http://www.firstmonday.org/issues/issue4_8/moglen/index.html#author

Moody, G. (2002). *Rebel code: Inside Linux and the open source revolution*. Oxford: Perseus.

Nissenbaum, H., & Howe, D. C. (2003). *Values in design*. Retrieved December 12 2003, from http://www.nyu.edu/projects/valuesindesign/index.html

Olson, M. (1971). *The logic of collective action: Public goods and the theory of groups*. London: Harvard University Press.

O'Reilly, T. (2001). *My definition of freedom zero*. Retrieved May 26 2003, from http://www.oreillynet.com/cs/weblog/view/wlg/526

O'Reilly, T. (2002). *The growing politicization of open source*. Retrieved June 25 2003, from http://www.oreillynet.com/pub/wlg/1840

O'Reilly, T. (2003). *"Freedom zero" and "Freedom or power": Tim O'Reilly and FSF leaders debate*. Retrieved June 25 2003, from http://linuxtoday.com/news_story.php3?ltsn = 2001-08-17-011-20-OP-CY-MS

Perens, B. (1999). *It's time to talk about free software again*. Retrieved June 16 2003, from http://lists.debian.org/debian-devel/1999/debian-devel-199902/msg01641.html

Perens, B. (2003). *The open source definition*. Retrieved June 21 2003, from http://www.opensource.org/docs/definition.html

Phillips, L., & Jørgensen, M. W. (2002). *Discourse analysis as theory and method*. London: Sage.

Post, E. (2003). *Real programmers don't use Pascal*. Retrieved June 10 2003, from http://www.edfac.usyd.edu.au/staff/souters/Humour/Real.Programmer.Stories.html

Python. (2003). *Python's development process*. Retrieved May 2 2003, from http://www.python.org/dev/process.html

Rand, A. (1992). *Atlas shrugged*. New York: Penguin.

Raymond, E. S. (1999a). *The magic cauldron*. Retrieved August 1 2003, from http://www.catb.org/~esr/writings/magic-cauldron/

Raymond, E. S. (1999b). *Why I am an anarchist*. Retrieved August 1 2003, from http://catb.org/~esr/writings/anarchist.html

Raymond, E. S. (2001a). *The cathedral and the bazaar: Musings on Linux and open source by an accidental revolutionary*. Cambridge: O'Reilly.

Raymond, E. S. (2001b). *Freedom, power, or confusion? [ESR on debate between O'Reilly and FSF]*. Retrieved June 25 2003, from http://linuxtoday.com/news_story.php3?ltsn = 2001-08-17-016-20-OP-CY

Raymond, E. S. (2001c). *Goodbye free software, hello open source*. Retrieved June 25 2003, from www.tuxedo.com/~esr

Raymond, E. S. (2002a). *Ethics from the barrel of a gun: What bearing weapons teaches about the good life*. Retrieved August 1 2003, from http://catb.org/~esr/guns/gun-ethics.html

Raymond, E. S. (2002b). *Why we fight – An anti-idiotarian manifesto*. Retrieved August 1 2003, from http://catb.org/~esr/aim/

Raymond, E. S. (2003a). *Homesteading the noosphere*. Retrieved June 10 2003, from http://www.firstmonday.dk/issues/issue3_10/raymond/

Raymond, E. S. (2003b). *The law is an ass*. Retrieved June 25 2003, from http://www.opendemocracy.net/debates/article.jsp?id = 8&debateId = 40&articleId = 246

Richardson, J. (2003). *Free software and GPL society: Interview with Stefan Merten of Oekonux*. Retrieved June 16 2003, from http://subsol.c3.hu/subsol_2/contributors0/mertentext.html

Sclove, R. E. (1995). *Democracy and technology*. London: Guildford Press.

Scoville, T. (1999). *Whence the source: Untangling the open source/free software debate*. Retrieved August 12 2003, from http://opensource.oreilly.com/news/scoville_0399.html

Slashdot. (2001a). *Eric S. Raymond answers*. Retrieved May 10 2003, from http://slashdot.org/interviews/99/09/30/1418244.shtml

Slashdot. (2001b). *Thus spake Stallman*. Retrieved May 10 2003, from http://slashdot.org/interviews/00/05/01/1052216.shtml

Slashdot. (2003). *Slashdot: News for nerds, stuff that matters*. Retrieved August 5 2003, from http://www.slashdot.org

Spectorsoft.com. (2003). *Home page*. Retrieved June 16 2003, from http://www.spectorsoft.com/

Stallman, R. M. (1991a). *Against software patents*. Retrieved June 25 2003, from http://lpf.ai.mit.edu/Patents/against-software-patents.html

Stallman, R. M. (1991b). *The GNU general purpose license*. Retrieved February 12 2003, from http://www.gnu.org/copyleft/gpl.html

Stallman, R. M. (1992). *Why software should be free*. Retrieved August 2 2003, from http://www.gnu.org/philosophy/shouldbefree.html

Stallman, R. M. (1993). *The GNU manifesto*. Retrieved February 10 2003, from http://www.gnu.org/gnu/manifesto.html

Stallman, R. M. (2002a). *Free software, free society: Selected essays of Richard M. Stallman*. Boston: GNU Press.

Stallman, R. M. (2002b). *RMS responds*. Retrieved August 2 2003, from http://www.slashdot.org

Stallman, R. M. (2003a). *Categories of free and non-free software*. Retrieved June 16 2003, from http://www.gnu.org/philosophy/categories.html#Non-CopyleftedFreeSoftware

Stallman, R. M. (2003b). *The free software definition*. Retrieved June16 2003, from http://www.gnu.org/philosophy/free-sw.html

Stallman, R. M. (2003c). *GNU is not Unix*. Retrieved February 10 2003, from http://www.gnu.org/

Stallman, R. M. (2003d). *Let's share! Richard Stallman*. Retrieved June 25 2003, from http://www.opendemocracy.net/debates/article-8-40-31.jsp

Stallman, R. M. (2003e). *Why free software is better than open source*. Retrieved June 6 2003, from http://www.gnu.org/philosophy/free-software-for-freedom.html

Stallman, R. M. (2003f). *Why software should not have owners*. Retrieved June 2 2003, from http://www.gnu.org/philosophy/why-free.html

Weber, S. (2003). *Open source software in developing economies*. Retrieved August 2 2003, from http://www.ssrc.org/programs/itic/publications/ITST_materials/webernote2.pdf

Weinstein, M. (2003). *Text analysis markup system* (Version 2.15a7). Retrieved 12 January 2003, from http://www.sourceforge.net/projects/tamsys

Williams, S. (2002). *Free as in freedom*. London: O'Reilly.

Winner, L. (1986). *The whale and the reactor: A search for limits in an age of high technology*. London: University of Chicago.

David M. Berry is an ESRC-funded DPhil research student at the University of Sussex, UK, in the Graduate Research Centre for Culture and Communication (CULCOM). His research is focused on the democratic potential of the Internet, particularly related to the Free/Libre and Open-Source movements and issues such as participatory and value-sensitive design and critical theories of technology. *Address*: University of Sussex, Media and Cultural Studies, Brighton. [email: d.berry@sussex.ac.uk]