International Journal of Business, Human and Social Sciences
ISSN: 2517-9411
Vol:10, No:7, 2016

Free and Open Source Licences, Software Programmers, and the Social Norm of Reciprocity

Luke McDonagh

***Abstract*—**Over the past three decades, free and open source software (FOSS) programmers have developed new, innovative and legally binding licences that have in turn enabled the creation of innumerable pieces of everyday software, including Linux, Mozilla Firefox and Open Office. That FOSS has been highly successful in competing with 'closed source software' (e.g. Microsoft Office) is now undeniable, but in noting this success, it is important to examine in detail why this system of FOSS has been so successful. One key reason is the existence of networks or communities of programmers, who are bound together by a key shared social norm of 'reciprocity'. At the same time, these FOSS networks are not unitary – they are highly diverse and there are large divergences of opinion between members regarding which licences are generally preferable: some members favour the flexible 'free' or 'no copyleft' licences, such as BSD and MIT, while other members favour the 'strong open' or 'strong copyleft' licences such as GPL. This paper argues that without both the existence of the shared norm of reciprocity and the diversity of licences, it is unlikely that the innovative legal framework provided by FOSS would have succeeded to the extent that it has.

***Keywords*—**Open source, software, licences, reciprocity, networks.

I. Introduction

A. Free and Open Source Software (FOSS) Licensing Systems

IN the online world, any alternative to the traditional copyright licensing system must endeavour to serve the underlying purposes of copyright while not restricting the workings of the internet. The subject of this article is one such alternative, the free and open source licensing system. The development of 'free' and 'open source' licensing has occurred over the past thirty years, during a period of intense debate about the role of intellectual property rights in the online era [1]. Indeed, within the current political and economic sphere, there is little consensus regarding the issue of how broad intellectual property protection should be with respect to new forms of computer software and operating systems. In this respect it is notable that FOSS programmers typically object to the traditional 'closed', and 'for-profit', licensing model of copyright [2]. This objection is typically founded upon the argument that intellectual property law fails to adequately facilitate the processes of cultural innovation within the context of software development [3]. For instance, adherents of FOSS tend to hold the belief that the 'closed code' model of software development is unduly restrictive and that it put too much power in the hands of corporate entities [4]. Crucially, many software programmers seek to develop their software works in a collaborative, online environment. Providing access to the source code is vital in order to allow programmers to improve and modify the software in accordance with their own needs. It is within this context that the first FOSS licences were developed [5]. The development of the licences was therefore a necessary achievement in order to further a novel, practical goal - to 'open' up software development to all potential contributors.

The well-known NGO 'the Open Source Initiative' has stated that 'open source' has the goals of facilitating free redistribution, allowing open access to the source code, allowing the distribution of derived works under the same terms, protecting the integrity of the author's source code, and preventing the licences from restricting other software programs. There are two fundamental principles of 'open source': (i) users ought to be able to use open source software for any purpose; (ii) users ought to be able to modify and distribute the software without the need for prior authorisation from the initial developer [5]. Nonetheless, Westkamp has noted that finding ways to regulate this ethos within the law is not a straightforward matter [6]. For this reason, as detailed further below, there are many different types of free and open source licences, and the terms of each licence depend upon whether the licence is a 'permissive' or 'restrictive' open source licence.

The FOSS system of licensing works alongside copyright, rather than replacing it. At a basic level, open source has been described as 'a legal construct for cooperation and trade in intellectual property' [7]. It takes advantage of the fact that in certain contexts some authors may not wish to avail of all of the rights which arise automatically under [8]. In this vein, Kelty has stated that alternative licensing systems 'rely on the existence of intellectual property' in order to maintain creativity e.g. within a network 'even as they occupy a position of challenge or resistance to the dominant forms of intellectual property' [9]. Similarly, Dixon has remarked that there is 'real ingenuity' concerning the way open source licences have the ability 'to both reinforce the prevailing legal regime of copyright and to undermine it' [10].

FOSS licence drafters attempt to 'bend' the traditional copyright licensing system so that it can be better tailored to suit the individual creator - the software programmer. In fact, it is only because of the underlying copyright protection that the authors/programmers have the capability to license the programs contractually. This is an innovative tactic, and it holds several advantages over the position of pursuing a legal amendment e.g. via lobbying. For one thing, it saves time,

Dr Luke McDonagh is with the City Law School, City University of London, London, UK (e-mail: luke.mdonagh@city.ac.uk).

money and effort on the part of the programmers. Furthermore, as detailed further below, the alternative licensing solution also can better satisfy a great diversity of views on the subject. Within the FOSS system a number of different licences can be created, whereas a single amendment to copyright might actually end up being overly permissive or unduly restrictive to the needs of some programmers.

In the case of a software program featuring 'open' source code, any programmer is free to access the code and to modify the program. In other words, keeping the source code 'open' facilitates collaborative creation. As detailed below, in order to enable 'open source' collaboration, individual authors choose to license the economic and moral rights over their works in a flexible way, with the goal of allowing other programmers the freedom to use, modify and improve the programs without fear of breaching the laws of copyright. Furthermore, as described below, there is a normative foundation underpinning the legal framework of open source licences. However, the terminology of FOSS is not settled. Within FOSS discourse there is also much discussion of the terminology of the licences themselves. To some extent there is a divide between supporters of 'free' licences and supporters of 'open' licences. The term 'copyleft' is used widely, but can mean different things in different contexts.

B. Exploring the Diversity of FOSS Licences

To date, around 100 FOSS licences have been approved by the Open Source Initiative, a non-governmental organisation which aims to make the world of open source licensing accessible and comprehensible both to programmers who seek to utilize the licences and to users of open source software. The open source licences themselves range from being 'permissive' licences featuring very few restrictions, such as the Apache licence and the Berkeley Software Distribution licence (BSD), to 'restrictive' licences, such as the GNU GPL version 3. There are also licences, such as the Mozilla Public Licence (MPL) and Eclipse, which aim to provide a balance between these two extremes. Microsoft has even drafted its oven open source licences, which have been approved by the Open Source Initiative (OSI). The European Union also now has its own official open source licence, the European Union Public Licence (EUPL).

The reason for the prevalence of different licences is that software programmers form a diverse group. Different groups of open source programmers may favour different licences. Some programmers tend to favour the 'permissive' licences, while other programmers favour the 'restrictive' licences. Furthermore, individual programmers may also undertake varying different types of activities. For instance, a programmer may engage in commercial activities with respect to some programs, while at the same time he or she may also work on other programs in a non-commercial capacity. For this reason, a programmer potentially has the freedom under the open source model to utilise one type of open source licence for commercial activities and another type of licence for non-commercial activities. For instance, if a software programmer intends to create a program, the programmer would like other programmers to be able to modify and improve it, he or she may find that a number of licences will allow this effectively. However, the need for distinct licences becomes clear if additional terms are required by the programmer. For example, if the programmer wishes to allow others to modify and improve the work, and he or she does not mind what the other programmers may do with the modified work, commercially or otherwise, a permissive-type licence such as the Apache licence or the Berkeley Software Distribution licence (BSD) would probably be the best option. These types of licence give complete freedom to the later user/ contributor/distributor regarding use and distribution of the work. The code can even be integrated into 'closed' programs under and distributed commercially under this type of licence. On the other hand, if the programmer wishes to prevent others from any commercial use of the modified, or 'derivative', work, and the programmer wants the code to remain open, a restrictive licence such as GPL version 3 would be preferable. Under GPL version 3, all derivative content and 'linked' works must feature open code and the programs cannot be distributed commercially. Other licences tend to fall in between these two extremes. An example of this is the Mozilla Public Licence (MPL), which allows any distributions of modified code to be kept 'closed', and therefore potentially more commercially viable, as long as the original program's source code remains 'open'. There is therefore a great diversity of OS licences. This enables programmers to pick and choose the terms of their licences.

As noted above, under the open source model, each individual programmer has the ability to license his or her authored programs in line with his or her own individual wishes. However, in a case where the programmer is seeking to make contributory modifications to an existing open source program, he or she may be restricted in distributing a modified version of the program under the terms of the original open source licence. If the open source licence under which the original was distributed was a 'permissive' licence, the new contributor will not face much by way of licence term. Typically, the contributor would only be required to make accessible the terms of the original licence. However, if the open source licence under which the original work was distributed was a 'restrictive' licence such as GPL version 3, then any modified code featured in the new version of the program may also be required to be left 'open'. The inability to keep even the new modifications 'closed' could discourage some contributors because it would potentially undermine commercial exploitation of the modified program. GPL version 3 has been criticised by some programmers for being too 'strict' for this reason. Nevertheless, all of this diversity may come at a cost. For instance, the proliferation of different licences that are potentially available to programmers may make it difficult for later users/contributors/distributors to comprehend which uses are acceptable and legal [11]. For this reason, the Open Source Initiative has tried to curb the enactment of new licences, and some older, or poorly designed, licences have effectively been' retired' from use. However, these efforts have largely failed to prevent the

negative aspects of proliferation from taking place [11]. Furthermore, the majority of software programs are released under one of the 10 most popular licences, which does to some extent mitigate some of the proliferation issues. On this point, it has been argued that the proliferation of licences represents both 'helpful diversity' and 'hopeless confusion' [11]. In other words, despite the potential for confusion, there may be no other way to satisfy the diverse licensing needs of the software programmers [12].

II. Social Norms

A. Social Norms Are Prevalent within FOSS Licensing

As previously noted, this article argues that underpinning the legal aspects of FOSS is a foundation of informal social norms [13]. On this point has been stated that the open source model has in recent years 'imposed itself as a powerful social ideology' [5]. The importance of informal social norms in this context ought not to be underestimated. Indeed, the presence of shared social norms has been crucial to the success of network-based creativity within the FOSS movement - indeed, the unifying principle of FOSS is 'built upon an ethos of sharing' [6]. As discussed below, this ethos of sharing can be described by the term 'generalised reciprocity'.

Belenzon and Schankerman have performed empirical research on the motivations of open source programmers [14]. They found that the ideology of 'open source' plays a crucial role in motivating programmers. They also found that reputational factors were significant, and to a limited extent, a direct form of reciprocity was also an influence on the motivations of programmers. Given its importance in this context, the ideology of 'open source' must be examined in a normative sense.

As noted above, central to the ideology of FOSS is the fact that FOSS programmers believe that 'software should be offered to users with open access to the source code' and that 'end-users should be freely able to modify, copy, or redistribute' the programs in order to create better and more accessible software [10]. In other words, this ideology is centred on a belief that software is best produced when the code is left 'open' so that all other programmers have the capacity to make contributions to improve the software. As noted above, this ideology is clearly based on some form of 'share-ethic'. This article describes the underlying normative base of open source as 'generalised reciprocity'. Iannacci has previously used this term to describe the motivational factors underpinning open source. He remarked [15]:

> "… open source developers do contribute their derivative works back to the community… this is the case because they share norms of generalised reciprocity whereby they are entitled to expect future reciprocation in the form of the derivative work of third parties."

This idea of 'generalised reciprocity' differs from the direct form of 'reciprocity' examined by Belenzon and Schankerman - they envisaged a scenario where one programmer would contribute to one project, and in return another programmer would reciprocate by contributing to the original programmer's own project [14]. As Belenzon and Schankerman stated, this form of direct reciprocity plays a strong role in a limited number of projects, but it generally does not play a large role in motivating open source programmers. However, this article argues that the underlying ideology of open source, which Belenzon and Schankerman acknowledge is a crucial motivational factor for open source programmers, does have a normative foundation based upon this broader concept of 'generalised reciprocity' [14]. Under this idea, programmers use and contribute to open source projects not because they believe that other programmes will necessarily reciprocate to their own individual projects, but because they believe that within the grand scheme of 'open source' they ought to share their contributions within the network, while simultaneously reaping the benefits of other shared contributions. It appears that open source programmers believe that this model ultimately produces better and more accessible software than the traditional 'closed', proprietary model. Furthermore, the existence of a large, global network of programmers helps to facilitate this process. On this point, it has been noted that the presence of interdependent relationships within the network is a key part of the success of open source [16]. In this vein, central to Benkler's vision of a 'network' is the concept of 'peer production' [17]. This idea envisages a number of programmers, or 'peers', contributing to the development of software in a collaborative environment. These programmers are not necessarily working together in a team in the same building, in the same city or even in the same country. Each programmer has the ability to make his or her own individual changes and modifications to the software and distribute this via the internet. Because the source code for the new version of the program is accessible, other programmers have the ability to continue make further changes to the program down the chain of 'peers'. Therefore, it appears that open source programmers form 'generalised' and 'indirect' reciprocal bonds with each other, with the internet providing the crucial conduit for this process [18]. Indeed, there are parallels to be found with other creative contexts. For instance, Loshin has recently discussed the notions of authorship, ownership and creativity within the magicians' community in the US, and in particular the crucial role played by social norms, rather than intellectual property law, in regulating the sharing of magic tricks [19]. In relation to 'stand-up comedy', it has been argued that until recent times, the transmission and performance of jokes by comedians were 'governed by an open access regime' which was also based largely on social norms [20]. Furthermore, it has also been noted that over the past few decades, in response to social and economic pressures, comedians have developed a more complex set of 'non-legal norms, institutions and practices that maintain a non-trivial set of incentives to create' [20]. Programmers within the open source network have taken this informal, normative approach to regulation a step further. Not only is 'generalised reciprocity' part of the ideology of open source, individuals within the network have also created a system of licences in order to legally and formally regulate the creation and distribution of open source software.

III. Conclusion

For the purpose of this article, two points are of significance. Firstly, the programmers have developed and encouraged the use of new, innovative and legally binding licences. These licences are formed of a complex mixture of copyright and contract law. Secondly, the success of this system of free and open-source licensing appears to be due to the existence of a network or community of programmers, who are bound together by some shared social norms which can be grouped under the general idea of 'generalised reciprocity'. The open source network is not unitary – it is a diverse network and there are divergences of opinion between members regarding which licences are generally preferable. Some members favour the 'free' licences and other members favour the 'strong open' licences. Each individual member of the community licenses his or her works depending on his or her own individual wishes. Nonetheless, without the existence of shared norms, it is unlikely that the innovative legal framework provided by the open source system would have succeeded to the extent that it has. In particular, the open source system would not have proved successful had the unifying principle of generalised reciprocity not been accepted broadly.

Overall, the open source system has been a great success, though some challenges remain, particularly due to licence proliferation and patent issues. Furthermore, the success of these types of licences for software programs is largely down to the programmers themselves. The programmers have created tailored licence-based solutions rather than resorting to the use of traditional modes of political law-making. Instead of lobbying for an amendment to the intellectual property statutes, the programmers have successfully taken the law into their own hands.

References

[1] D. Hunter, 'Culture War,' *Tex L Rev* 83 (2004-2005), 1105.
[2] L. Rosen, *Open Source Licensing – Software Freedom and Intellectual Property Law* (Upper Saddle River, NJ: Prentice Hall PTR, 2004), 1-18.
[3] M. Henley & R. Kemp, 'Open Source Software: An introduction,' *Computer Law & Security Review* 24 (2008), 77, 77-78.
[4] R. Stallman, *Free Software, Free Society* (GNU Press, 2002), 47-48.
[5] L. Guibault and O. van Daalen, *Unravelling the Myth around Open Source Licences: An Analysis from a Dutch and European Law Perspective* (The Netherlands: T.M.C. Asser Press, 2006), 6-11.
[6] G. Westkamp, 'The Limits of Open Source' *IPQ* (2008), 14, 57.
[7] V. Lindberg, *Intellectual Property and Open Source: A Practical Guide to Protecting Code* (O'Reilly Media, 2008), 155.
[8] G. P. Rosloff, '"Some Rights Reserved": Finding the Space Between All Rights Reserved and the Public Domain,' *Columbia Journal of Law & the Arts* 33 (2009), 37, 68.
[9] C. M. Kelty, 'Punt to Culture,' *Anthropological Quarterly* 77 (2004), 547, 547.
[10] R. Dixon, *Open Source Software Law* (Artech House, 2004), 22.
[11] R. W. Gomulkiewicz, 'Open Source License Proliferation: Helpful Diversity or Hopeless Confusion?' *Washington University Journal of Law and Policy* 30 (2009), 261, 261-263.
[12] M. F. Schultz, 'Copynorms: Copyright Law and Social Norms,' (2006), 1. 1-5 - http://papers.ssrn.com/sol3/papers.cfm?abstract_id=933656. E. Ostrom, *Governing the Commons* (Cambridge: Cambridge University Press, 1990).
[13] R. P. Merges, 'Individual Creators in the Cultural Commons,' *Cornell Law Review* 95 (2010), 793, 805.
[14] S. Belenzon and M. Schankerman, 'Mark Motivation and sorting in open source software innovation,' *EDS Discussion Papers* DP019 (2008), 1, 4-5.
[15] F. Iannacci, 'The social epistemology of open source software development: The Linux case study,' (2005), 1, 11 - https://www.researchgate.net/publication/267259227_THE_SOCIAL_EPISTEMOLOGY_OF_OPEN_SOURCE_SOFTWARE_DEVELOPMENT_THE_LINUX_CASE_STUDY
[16] A-K Kuehnel, 'Microsoft, Open Source and the software ecosystem: of predators and prey—the leopard can change its spots,' *Information & Communications Technology Law* 17 (2008), 107, 109-110.
[17] Y. Benkler, *The Wealth of Networks* (New Haven, CT: Yale University Press, 2006), 59-91.
[18] J. Feller and B. Fitzgerald, *Understanding Open Source Software Development* (Addison Wesley, 2002), 1-5.
[19] J. Loshin, 'Secrets Revealed: How Magicians Protect Intellectual Property Without Law,' (2007), 1, 1-13 - http://www.udel.edu/richard/cisc356/2010/SSRN-id1005564.pdf
[20] D. Oliar and C. J. Sprigman, 'There's No Free Laugh (Anymore): The Emergence of Intellectual Property Norms and the Transformation of Stand-Up Comedy,' *Virginia Law Review* 94 (2008), 1787, 1865-1866.