MANTECH Publications

Journal of Software Engineering & Software Testing
Volume 7 Issue 2
ISSN: 2457-0516 (Online)

A Study on Free and Open Source Software (FOSS) and the Social Norm of Reciprocity

Akansha Goyal[1], Geetanjali Bansal[2]

Assistant Professor[1], Student[2]

Department of CSE

Arvind Gavali College of Engineering

Corresponding Author's Email: - *bansalgeetanjali21@yahoo.com[2]*

Abstract

Free and open source software (FOSS) programmers have created new, inventive, and legally binding licences during the last three decades, allowing the production of many pieces of daily software such as Linux, Mozilla Firefox, and Open Office. It is clearly apparent that FOSS has been very effective in competing with 'closed source software' (e.g., Microsoft Office), but in acknowledging this achievement, it is vital to investigate in depth why this FOSS system has been so successful. One important explanation is the development of networks or communities of programmers who are linked by a major common social norm of 'reciprocity.'

At the same time, these FOSS networks are not homogeneous; they are highly diverse, and there are significant differences of opinion among members about which licences are generally preferable: some members prefer the flexible 'free' or 'no copyleft' licences, such as BSD and MIT, while others prefer the 'strong open' or' strong copyleft' licences, such as GPL. This study contends that without both the shared norm of reciprocity and the diversity of licences, the new legal framework given by FOSS would not have flourished to the level that it has.

Keywords: *Networks, Licence, Reciprocity, Open Source, Software*

Page 76-84 © MANTECH PUBLICATIONS 2022. All Rights Reserved

INTRODUCTION

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

In the online world, any option to the traditional copyright licensing system must strive to serve the understated purpose of copyright while not restricting the workings one such option is the free and open source licencing system, which is the focus of this article. Over the last thirty years, amid a period of strong discussion over the role of intellectual property rights in the internet era [1,] the creation of 'free' and 'open source' licencing has happened. Indeed, there is no agreement in the present political and economic sector on how far intellectual property protection should be extended to new kinds of computer software and operating systems. In this regard, it is worth noting that FOSS programmers often oppose the traditional 'closed' and 'for-profit' copyright licencing paradigm [2]. This complaint is often based on the claim that intellectual property legislation fails to sufficiently promote cultural innovation processes in the context of software development [3]. For example, supporters of FOSS believe that the 'closed source' approach of software development is overly restrictive and concentrates too much power in the hands of corporate corporations [4]. Most importantly, many software developers want to create their product in a collaborative, online setting. Access to the source code is critical for allowing programmers to update and alter the software to meet their own demands. The first free and open source licences were created in this context [5]. The creation of the licences was thus a vital step toward achieving a fresh, realistic goal: to 'open' up software development to all possible contributors.

According to the well-known NGO 'the Open Source Initiative,' the goals of 'open source' are to facilitate free redistribution, allow open access to the source code, allow the distribution of derived works under the same terms, protect the integrity of the author's source code, and prevent licences from restricting other software programmes. There are two key concepts of 'open source': I users should be allowed to use open source software for any purpose; and (ii) users should be able to alter and distribute the programme without prior permission from the original creator [5]. Nonetheless, Westkamp has observed that finding means to govern this attitude inside the law is not easy [6]. As a result, as explained further below, there are several varieties of free and open source licences, and the conditions of each vary depending on whether the licence is a

Page 76-84 © MANTECH PUBLICATIONS 2022. All Rights Reserved

'permissive' or a'restrictive' open source licence.

The FOSS licence scheme complements, rather than replaces, copyright. At its most fundamental, open source has been defined as "a legal foundation enabling intellectual property cooperation and commerce" [7]. It takes use of the reality that, under some circumstances, certain authors may not choose to exercise all of the rights conferred by [8]. In this line, Kelty claims that alternative licencing systems "rely on the presence of intellectual property" to preserve innovation, for example, inside a network, "even as they occupy a position of challenge or opposition to the prevailing forms of intellectual property" [9]. Similarly, Dixon has stated that there is "genuine inventiveness" in the way open source licences have the power "to simultaneously strengthen and destroy the dominant legal framework of copyright."

FOSS licence drafters aim to 'bend' the standard copyright licencing structure in order to better fit the individual author - the software programmer. In actuality, the authors/programmers may only licence the programmes contractually because of the underlying copyright protection. This is a new strategy that has various advantages over the traditional approach of pursuing a law modification, for example, by lobbying. For one thing, it saves the programmers time, money, and effort. Furthermore, as discussed further below, the alternative licence arrangement can better meet a wide range of viewpoints on the matter. A variety of distinct licences may be produced inside the FOSS system, but a single change to copyright may wind up being either permissive or extremely restrictive to the demands of particular programmers.

In the case of a software application with "open" source code, any programmer is allowed to access and edit the code. In other words, having the source code "open" encourages collaborative development. Individual writers opt to licence the economic and moral rights over their works in a flexible manner in order to provide other programmers the flexibility to use, change, and develop the programmes without concern of violating copyright rules, as stated below. Furthermore, as stated below, the legal architecture of open source licences is supported by a normative foundation. However, there is no consensus on FOSS nomenclature. There is also a lot of debate about the vocabulary of the licences themselves among the FOSS community. There is some disagreement between

Page 76-84 © MANTECH PUBLICATIONS 2022. All Rights Reserved

proponents of 'free' licences and supporters of 'open' licences. The phrase "copyleft" is extensively used, yet it can refer to different things in different circumstances.

B. Exploring the Diversity of FOSS Licences

To date, the Open Source Initiative has approved approximately 100 FOSS licences. The Open Source Initiative is a non-governmental organisation that aims to make the world of open source licencing accessible and understandable to both programmers seeking to use the licences and users of open source software. The open source licences themselves range from 'permissive' licences with few limitations, such as the Apache licence and the Berkeley Software Distribution licence (BSD), to'restrictive' licences, such as the GNU GPL version 3. There are some licences that try to strike a balance between these two extremes, such as the Mozilla Public Licence (MPL) and Eclipse. Microsoft has even written open source licences for their ovens, which have been accepted by the Open Source Initiative (OSI). The European Union has its own official open source licence, known as the European Union Public Licence (EUPL).

The predominance of different licences is due to the fact that software programmers are a heterogeneous community. Different groups of open source programmers may choose various licences. Some programmers prefer the 'permissive' licences, while others prefer the'restrictive' licences. Furthermore, individual programmers may engage in a wide range of tasks. A programmer, for example, may engage in commercial activities with regard to some projects while concurrently working on other programmes in a non-commercial capacity. As a result, under the open source paradigm, a programmer may use one type of open source licence for commercial operations and another type of licence for non-commercial activities. For example, if a software programmer plans to write a programme and would like other programmers to be able to edit and enhance it, he or she may discover that a number of licences would effectively allow this. If the programmer requires extra terms, the requirement for several licences becomes evident. For example, if the programmer wants to allow others to modify and improve the work and doesn't mind what the other programmers do with the modified work, commercially or otherwise, a permissive-type licence, such as the Apache licence or the Berkeley Software Distribution licence

Page 76-84 © MANTECH PUBLICATIONS 2022. All Rights Reserved

(BSD), is probably the best option. These sorts of licences offer the subsequent user/contributor/distributor total control over the use and dissemination of the work. Under this form of licence, the code can even be incorporated into 'closed' applications and marketed commercially. If, on the other hand, the programmer desires to prevent others from making commercial use of the modified, or 'derivative,' work while keeping the source open, a restrictive licence such as GPL version 3 is preferred. All derivative content and 'linked' works must include open source under GPL version 3, and the programmes cannot be distributed commercially. Other licences often lie between these two extremes. The Mozilla Public Licence (MPL) is an example of this, since it permits any distributions of changed code to be maintained 'closed,' and hence potentially more economically viable, as long as the original program's source code remains 'open.' As a result, there is a wide range of OS licences. This allows programmers to customise the terms of their licences.

As previously stated, under the open source approach, each individual programmer has the opportunity to licence his or her produced programmes in accordance with his or her own personal preferences. However, if the programmer wishes to contribute to an existing open source software, he or she may be prohibited from sharing a modified version of the programme under the conditions of the original open source licence. If the original was provided under a 'permissive' open source licence, the new contributor will not face many licencing issues. Typically, the donor would just be obliged to make the conditions of the original licence available. However, if the original work was published under a'restrictive' licence, such as GPL version 3, any changed code included in the new version of the software may also be required to be left 'open'. The inability to keep even fresh modifications 'closed' may deter some contributors since it may jeopardise commercial usage of the changed application. Some programmers have criticised GPL version 3 for being too'strict' for this reason. Nonetheless, all of this variety may come at a price. For example, the plethora of alternative licences accessible to programmers may make it difficult for subsequent users/contributors/distributors to understand which uses are acceptable and lawful [11]. As a result, the Open Source Initiative has attempted to limit the introduction of new licences, and some older, or poorly structured, licences have

Page 76-84 © MANTECH PUBLICATIONS 2022. All Rights Reserved

essentially been "retired." However, these measures have mostly failed to mitigate the negative consequences of proliferation [11]. Furthermore, the vast majority of software products are distributed under one of the top ten most prevalent licences, which helps to reduce some of the proliferation difficulties. It has been suggested that the proliferation of licences constitutes both "useful diversity" and "hopeless confusion" [11]. In other words, despite the risk of misunderstanding, there may be no alternative way to meet the different licencing requirements of software developers [12].

SOCIAL NORMS

A. Social Norms Are Prevalent within FOSS Licensing

As previously said, this paper contends that informal social norms support the legal features of FOSS [13]. On this topic, it has been argued that the open source approach has "imposed itself as a significant social ideology" in recent years [5]. The significance of informal social norms in this situation should not be overlooked. Indeed, the prevalence of common social norms within the FOSS movement has been critical to the development of network-based innovation - indeed, the unifying concept of FOSS is 'built around an ethos of sharing' [6]. This shared mentality can be characterised by the phrase 'generalised reciprocity,' as explained more below.

Belenzon and Schankerman conducted empirical study on open source programmers' motives [14]. They discovered that the 'open source' mindset is vital in encouraging programmers. They also discovered that reputational reasons were crucial, and to a lesser extent, a direct form of reciprocity had an impact on programmers' motives. Because of its significance in this setting, the ideology of "open source" must be analysed normatively.

As previously said, FOSS programmers think that "software should be supplied to users with open access to the source code" and that "end-users should be freely allowed to edit, copy, or redistribute" the programmes to produce better and more accessible software [10]. In other words, this ideology is based on the concept that software is best generated when the code is left 'open,' allowing all other programmers to contribute to its improvement. As previously said, this worldview is definitely founded on some type of'share-ethic.' The basic moral foundation of open source is described as 'generalised reciprocity' in this article.

Page 76-84 © MANTECH PUBLICATIONS 2022. All Rights Reserved

This word has already been used by Iannacci to characterise the motivating reasons underlying open source. He stated [15]: "... open source developers do submit derivative works to the community..." This is because they share generalised reciprocity rules, which allow them to expect future reciprocation in the form of derivative work from third parties."

This concept of 'generalised reciprocity' differs from the direct form of'reciprocity' investigated by Belenzon and Schankerman, who imagined a scenario in which one programmer contributed to one project, and another programmer reciprocated by contributing to the original programmer's own project [14]. As mentioned by Belenzon and Schankerman, this type of direct reciprocity is important in a small number of projects, but it does not play a big role in encouraging open source programmers in general. This paper contends, however, that the underlying ideology of open source, which Belenzon and Schankerman concede is a critical motivator for open source programmers, does have a normative grounding based on this larger idea of 'generalised reciprocity' [14]. According to this theory, programmers use and contribute to open source projects not because they expect other programmes to reciprocate to their own individual projects, but because they believe that in the grand scheme of 'open source,' they should share their contributions within the network while reaping the benefits of other shared contributions. Open source programmers tend to believe that this paradigm will eventually provide better and more accessible software than the previous 'closed', proprietary one. Furthermore, the availability of a vast worldwide network of programmers aids in this process. On this issue, it has been said that the presence of interdependent interactions inside the network is a critical component of open source success [16]. In this manner, the notion of 'peer production' [17] is key to Benkler's vision of a 'network.' This concept envisions a group of programmers, or 'peers,' contributing to the creation of software in a collaborative environment; these programmers are not necessary working in the same building, city, or even nation. Each coder can make his or her own tweaks and adjustments to the software and share them through the internet. Because the source code for the new version of the software is available, other programmers can continue to make changes to the programme farther down the chain of 'peers.' As a result, it appears that open source programmers build

Page 76-84 © MANTECH PUBLICATIONS 2022. All Rights Reserved

"generalised" and "indirect" reciprocal ties with one another, with the internet serving as a critical channel for this process [18]. There are indeed parallels with various creative environments. Loshin, for example, has lately examined the concepts of authorship, ownership, and creativity within the magicians' community in the United States, emphasising the critical importance of social norms, rather than intellectual property legislation, in controlling the sharing of magic tricks [19]. It has been suggested that, until recently, the transmission and delivery of jokes by comedians were "controlled by an open access system" that was also "primarily dependent on societal standards" [20]. Additionally, in response to social and economic constraints, comedians have established a more complicated network of 'non-legal conventions, institutions, and practises that preserve a non-trivial set of incentives to produce' [20]. Open source programmers have taken this informal, normative approach to regulating a step further. Not only is 'generalised reciprocity' part of the open source idea, but members of the network have also developed a licencing structure to legally and officially govern the production and distribution of open source software.

CONCLUSION

Two considerations are important for the purposes of this paper. To begin, the programmers created and promoted the usage of new, creative, and legally binding licences. These licences are a complicated blend of copyright and contract law. Second, the success of this free and open-source licencing structure appears to be attributable to the development of a network or community of programmers who are held together by some common social norms that may be classified under the broad concept of 'generalised reciprocity.' The open source network is not a unified one; it is heterogeneous, and members have differing views on which licences are typically desirable. Some members prefer 'free' licences, while others prefer'strong open' licences. Each individual member of the community licences his or her works based on his or her own personal preferences. Nonetheless, without the establishment of common standards, it is unclear that the open source system's novel legal structure would have thrived to the level that it has. The open source system, in particular, would not have been effective if the unifying idea of generalised reciprocity had not been widely embraced.

Page 76-84 © MANTECH PUBLICATIONS 2022. All Rights Reserved

Overall, the open source system has been a huge success, but there are still some concerns to work out, notably with licence proliferation and patent issues. Furthermore, the success of these sorts of software licences is primarily dependent on the programmers themselves. Instead of using traditional ways of political law-making, the programmers devised customised license-based solutions. Instead than campaigning for changes to intellectual property laws, programmers have successfully taken the law into their own hands.

REFERENCES

1. C. M. Kelty, 'Punt to Culture,' Anthropological Quarterly 77 (2004), 547, 547.
2. 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).
3. M. Henley & R. Kemp, 'Open Source Software: An introduction,' Computer Law & Security Review 24 (2008), 77, 77-78.
4. 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.
5. R. Dixon, Open Source Software Law (Artech House, 2004), 22.
6. R. Stallman, Free Software, Free Society (GNU Press, 2002), 47-48.
7. G. Westkamp, 'The Limits of Open Source' IPQ (2008), 14, 57.
8. R. P. Merges, 'Individual Creators in the Cultural Commons,' Cornell Law Review 95 (2010), 793, 805.
9. L. Rosen, Open Source Licensing – Software Freedom and Intellectual Property Law (Upper Saddle River, NJ: Prentice Hall PTR, 2004), 1-18.
10. R. Stallman, Free Software, Free Society (GNU Press, 2002), 47-48.

Page 76-84 © MANTECH PUBLICATIONS 2022. All Rights Reserved