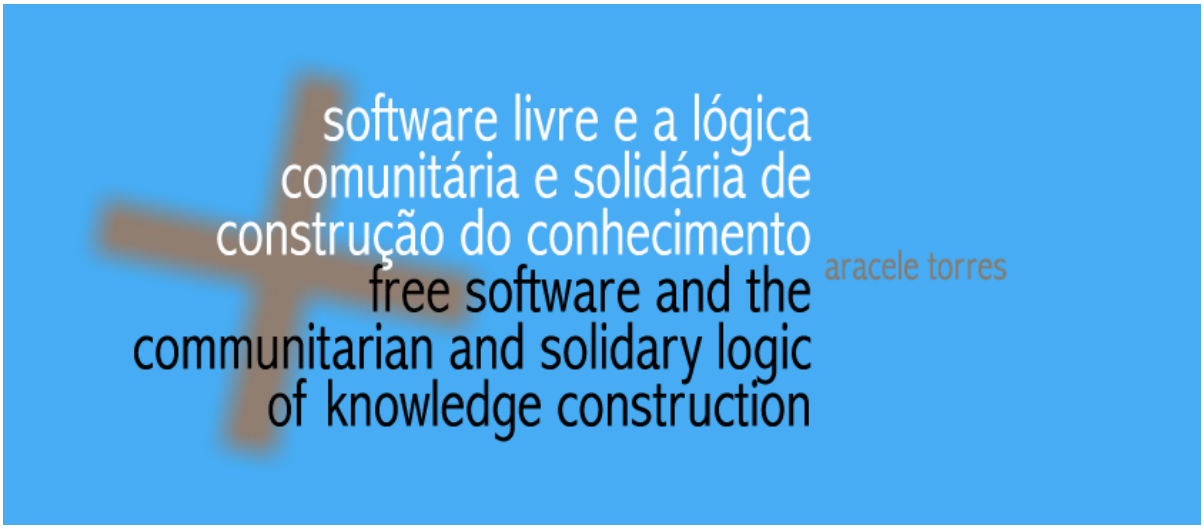
V!18
issn 2175-974x | ano 2019 year
semestre 01 semester



Aracele Torres is a Historian and Doctor in Social History. She works in the area of History of Science and Technology, developing research on the history of digital technology, free software, the Internet, cyber-libertarianism, ideology, and utopia.

Torres, A. L., 2019. Free software and the communitarian and solidary logic of knowledge construction. *V!rus*, Sao Carlos, 18. Translated from Portuguese by Juliano Veraldo da Costa Pita. [e-journal] [online] Available at: .

Abstract

This article briefly discusses how the notion of sharing and collective creation present in the free software production model is linked to an idea of law and social responsibility. We demonstrate how, through the creation of the GNU Project, which initiated the free software movement, Richard Stallman sought to oppose the anti-community and anti-solidarity logic imposed by the software industry from the end of the last century. This logic, when transforming the software into private goods of restricted use and circulation by the copyrights privileged individual action, creation and gain to the detriment to the collective's. Contrary to this trend, the free software movement presents itself as a defense not only of free knowledge as an important value in society, but also of its production in a collaborative way as essential for the free software and society as a whole to thrive.

Keyword Free Software, Collective Work, Collaboration, Social Responsibility

GUEST AUTHOR ARTICLE

1 Introduction

The open and collaborative production model of the GNU Project (GNU's Not Unix!)[1], which started the free software movement is a milestone in the history of modern computing and also represents a crack in the neoliberal individualistic production logic. This model proposes that computer programs keep their source code[2] open to anyone who wishes to execute them for any purpose; to distribute copies of these programs to anyone; to study them to understand their functioning and to be able to change them; and to redistribute copies of modified versions of those programs. These are the famous four basic freedoms of free software, which are related to their use permissions and not to their price. Free software is not necessarily without cost or non-commercial[3]. These freedoms not only allow as also legally enforce, through the use of the GPL (General Public License)[4]license, that the software is to be treated as a collective creation, which is in constant process of improvement and that allows the participation and collaboration of everyone.

The GPL license, also known as copyleft[5], obliges those who wish to redistribute the modifications made in free software to also use the same free license adopted by the original author of the software. This ensures that no one can take a open source free software, make changes to it, and redistribute the program with that code closed. This is a feature of the GPL that one of the executives of Microsoft once called, in a pejorative tone, "viral", but that is actually the secret of the success of the free software projects. The GPL uses the copyright mechanism itself - which it strongly opposes - to circumvent it and to reverse its logic, making it rather than restricting to copying and redistribution, as to act as a constraint on that restriction (Torres, 2018, p.151). If I write software and license it under the GPL, which is basically a copyright, I am telling the world that I would like everyone to have access to my software code, and that if someone decides to change it and redistribute it, this person would also be required to do the same.

The fact that the program code is open and allow for constant modifications and improvements makes free software a collective construction. One of the main consequences of this was the development of communities around specific free software projects, such as the KDE, Debian, Firefox and VLC communities. The dynamics of the contributions in these communities vary according to different factors, for example, the governance guidelines of each one, their size, the political-philosophical orientations of their members. It is very common for the people involved in them, the so-called collaborators or contributors, to volunteer and devote their time to contribute to these projects without demanding any financial contribution. But it is also common for companies or non-profit institutions, such as the Mozilla Foundation, responsible for the Firefox browser, to pay people to work on the maintenance of the project. And while that happens, there are still thousands of contributors around the world who volunteer in the development and maintenance of Firefox.

This model of production and distribution of free software, however, is the opposite of that practiced by the industry today, which has adopted, at least since the 1980s, the proprietary or closed source software model. In this model, the company or individual creates the software and uses legal mechanisms of intellectual property, usually copyright, to make its source code closed - that is, inaccessible to any user who wishes to understand how that program was constructed and how it functions. In this case, only the copyright holder has the legal right to access and change this code, as well as to distribute copies of such programs. If the user attempts to do any of these things, without permission, he is framed as a violator of an intellectual property system, a practice that is criminalized and pejoratively called "piracy" by the industry.

What prevailed as the industry standard was therefore an approach as the software as a private work or good, essentially with a commercial purpose, a product that needs to have its circulation and use controlled, as this directly impacts on the profit that its owners can have with it. In opposition to this is the view of the free software, in that a computer program should be seen as knowledge, and, therefore, a work of collective use and construction, a “commons”, which must have, first of all, social purposes. This does not mean, however, that it can not be marketed, but that it has a social function that must benefit everyone, not just an economic elite. Although it has become a heterogeneous[6] movement over time, it was born as a social movement in defense of free knowledge in society.

2 The emergence of proprietary software

The modern concept of “commons” was introduced in the academic literature in 1968 by ecologist Garrett Hardin in his famous article "The Tragedy of Commons," which dealt with the management of a finite resource commonly used by individuals acting in accordance with their own interests. In the sense it uses, "commons" refers to shared natural resources, such as rivers and lands, which in their opinion always tend to be exhausted for lack of rational use by individuals. The original use of this term refers to the context of medieval Britain and was used by British law to describe land commonly used for peasants and nobles. In the 1990s, the concept was appropriated by economics, especially after Elinor Ostrom's publication of Governing the Commons: The Evolution of Institutions for Collective Action, which opposed Hardin's thesis that the “commons” always tended to failure. Currently, the concept has been widely used to refer to digital cultural resources (intangible or intangible goods) that are produced and used collectively, such as Wikipedia or free software.[7]

Public goods, defined by neoclassical economic theory, are non-disputable goods, also called non-rival goods. Non-rivals, because their use by someone does not prevent the use by others. A public good also has the characteristic of non-exclusivity, that is, it is not possible to prevent someone from consuming it. In other words, they can be obtained without the need for a direct payment for their use. In opposition to this, rival and exclusive goods are considered private goods. However, non-exclusivity is not an inherent attribute of public goods, it is an attribute of a political character, related to the institutions that regulate its use. In general, there are almost no assets that can not be privatized through legal and political decisions. In fact, this process of privatizing knowledge, for example, is part of the value-extraction logic of contemporary capitalism (Prado, 2005).

The adoption of proprietary software as the industry standard was therefore a legal and political decision, a measure taken in a specific context of capitalism restructuration and the advance of neoliberalism in the 1980s. This restructuring led to a process of commoditization of intangible goods, such as science and technology, which have been artificially transformed into rival and exclusive goods. The way out of the capitalist structural crisis of the end of the last century, which culminated in a fall in profit and productivity, was the creation of new forms of capital accumulation. Technology and science were used both to increase the efficiency of the productive process (and thus profit), and as a product of this process itself (Dumeen and Levin, 2003). Capitalism revitalized itself by ensuring that these common goods were transformed into private property (Harvey, 2003). Similar to what was described by Karl Marx (2013) in the process of primitive capital accumulation, which occurred in Britain in the fifteenth century where lands of common use were transformed into private property of the nobility, a movement of closure of common goods occurred at the end of last century.

This phenomenon, called by David Harvey (2003) of "accumulation by dispossession," deals with the transformation of cultural, historical, and intellectual goods into commodities. Through this process, companies produce and sell these goods mainly in a model of rent, where the buyer pays for the right to use that product, but does not become exactly its owner, that is, it can not do what it wishes with it. Companies, therefore, exercise a monopoly on these goods, as is the case of closed source software. And this monopoly is secured by intellectual property laws, created with the justification of "protecting" the author's rights and "encouraging" the creation and dissemination of works, but which in fact have the function of creating an "artificial shortage" of such assets.

When Richard Stallman, the American programmer who started the free software movement, announced in 1983 his idea of creating a completely free operating system, GNU, built openly and collaboratively, it did not necessarily was starting a new trend. This was because it was common until the 1970s to share the software code in the universities and business environments. However, he was initiating a dispute that would secure an important place in the computer industry for this production model[8], as it fostered the development of an entire copyleft culture that goes beyond computation and is nowadays adopted by other areas such as music, the arts, science, etc. (Kelty, 2008). In his famous e-mail announcing the GNU Project, he uses as a fundamental argument for his undertaking: the notion that doing so was a moral duty to the community of programmers of which he was a part and to society itself as a whole. He was making a commitment not to let this collaborative culture, "as old as computers," as he himself (Stallman, 2002, p.17) said, die and be replaced by the proprietary, closed pattern we have in the industry today.

Until the 1970s, program code sharing was common practice in both corporate and university settings. Computing at that time was still very hardware-centric and the software industry was still in its infancy. There was no consensus among companies that one could profit from the software as much as possible with the hardware. From the 1970s to the 1980s the scenario, however, changes, with, among other things, the creation of personal computers; the creation of the first courses in Computer Science and the consequent supply of skilled labor; the development of programming languages; the advance of a neoliberal discourse defending market expansion. So companies start to see in the software a possibility of profit and rethink the approach around it, starting to adopt copyright to restrict its circulation and use and thus be able to profit from its sales. Before that, they usually provided the software along with the hardware, at no additional cost, and did not see much of a problem in allowing users to change the codes of those programs and share them (Torres, 2018).

It was this scenario of closure of the software that the free software project created by Richard Stallman opposed. Stallman wished to keep the software not only as a commodity of common use, but also of collective production. The logic was to establish a kind of ecosystem of collaboration, where the idea of only taking advantage of the work of another person (s) was substituted by the idea of to work with thi(e)s(e) author(s). An ecosystem where it is possible to be part of the creation of this work that is the software and to have a social responsibility in its maintenance as an open and collective work, belonging to all. This is a notion that goes against what neoliberalism prescribes, which preaches an anti-communitarianism in that it encourages the individual to work in isolation from the community, and defends the discourse that the individual is a self-made man, that wins alone in society and not necessarily in communion and cooperation with others, but in competition with them.

In the context of free software, the author has a social responsibility to the community in which he is inserted, sharing knowledge is a "golden rule", as Stallman put it when announcing his project (2002, p.31). In the neoliberal model, the responsibility of the author is for himself, not necessarily due to the community in which he is inserted, since what prevails is the idea of meritocracy. If he acquired some knowledge, a know-how, it was through his individual effort and he has every right to monopolize access to that knowledge and take advantage of it, since in neoliberal discourse private property is the natural right of the individual. There is no prospect of "social solidarity" or a "spirit of community", terms much used by Stallman in of your project.

3 Sharing as a right and a social responsibility

Richard Stallman argues that we have the right to make use of digital technologies in socially useful ways, taking advantage of the benefits offered by the ease of copying digital files. Digital information sharing is very different from sharing material things. While immaterial goods, such as a digital book or program code, are non-rival goods, material goods are the opposite. If someone has a printed book and wants to lend it to a friend, only one of them can have this book at that time. On the other hand, if this book is digital both can have it and read it at the same time. Since software is a knowledge, Stallman believes that this should be available for use by everyone. If someone owns a computer program and wants to share it, who should have the right to decide whether or not this sharing can be done: the individuals involved or the copyright holders of the program? For Stallman, in today's society this response is based on the economic criterion of profit maximization of software owner, usually a company. When, in fact, it should be based on another criterion: the prosperity that is achieved through the sharing of knowledge. In his opinion, when you are prevented from sharing knowledge, the whole society is impaired.

Although digital technologies facilitate the reproduction of information, copyright or any other form of restriction imposed by the intellectual property system, however, it moves in the opposite direction, blocking that facility. Copyright works from the idea that there is an owner, who has a monopoly over the copying and distribution of a work and is the one who legally decides the use that can be made of it. The interests of this owner do not always coincide with the interests of society in general. The act of restricting access to a work by means of copyright reflects the intention to produce an artificial shortage of this work. There is a conflict of interests, a tension between a social right, the free access to knowledge that Stallman believes we have, and the privilege of a monopolist to restrict that access for profit.

The distinction between the copyright holder and the author of the code is very important here because it clarifies the working and power relationship within this proprietary code production structure. The author of a software code, in general, is the worker (in this case, the person who programs) who writes it, and not the company or entrepreneur who sells it and profits from it. There is in the process of producing proprietary software the appropriation of a work from others. The programmer has the knowledge to produce software, but not the rights to use or reproduce that software, which is alienated from the employer who hired them. In the case of the production of free software, this logic can be broken, since the author of a free code still holds the rights to use and distribute that software that he wrote. Its work continue to be his, but he expands these rights to all of society, making software a common good that everyone can also enjoy.

Richard Stallman believes that the interests of society as a whole must be above private interests and that much importance is given to the author in contemporary society. According to his view, this importance results in a loss in the development of collective and collaborative production of knowledge, and the rights of the author are placed above the rights of the whole society. To emphasize that this importance is a product of our time and the economic system in which we live, he reinforces the idea that the most important factor in a society is the social production of knowledge, translated from the idea that a work must serve primarily social purposes, rather than the individual purposes of the one who created it.

Stallman explains that alongside the argument that the author has a natural right over his work - a right that would go beyond the social right of everyone to access to knowledge - the software industry also uses an economic argument to justify its monopoly on the source code. This argument can be summed up from the idea that if we do not produce proprietary software, the software will be terminated, because the developers would not work without the assurance that they would be rewarded. The mistake of this discourse, he says, is to assume that there are no alternative systems to proprietary software. In addition, it is also a great misconception to relate the existence of software, or any knowledge produced by society, to the idea of ownership. Software and private property are not synonymous, as he points out, "it is a consequence of the choice of socio-legal policy that we are questioning: the decision to have owners" (2002, p.122).

The crucial point to Richard Stallman and the proponents of the GNU Project is that society needs full access to knowledge, which is a right of all citizens. This defense is based on the "collectivist" view of the knowledge that project advocates have. Knowledge is seen as a common property and anything that arises in society, as new software, for example, derives from a common tradition. In this sense, the free software movement is today an important battlefield to guarantee the right of every citizen to free access and free sharing of information. For this movement, the adoption of an open system and the shared software model would therefore reflect a right and a social responsibility that is above individual capitalist interests.

4 Conclusion

By proposing a society in which all software is free, the GNU Project forces a reorientation of power and knowledge within the current social configuration, where knowledge in the form of science and technology is

one of the main sources of profit. This reorientation occurs by the breakdown of the monopoly of individuals or companies on a common good, in this case the technological knowledge. Both the access to this knowledge made free and its reproduction can not be controlled by mechanisms of privatization of the industry. Free software breaks with this logic of monopolization and also dilutes the power exercised over knowledge among all citizens, ensuring that software is really a public good.

The free sharing and collective and cooperative work that he consequently encourages with his production model are social responsibilities in the logic of movement. This logic rests on the idea of maintaining a logic of communal dynamic and solidarity that is opposed to the individualist and competitive neoliberal capitalist logic. In this sense, free software represents a shake in the fabric of capitalist discourse and shows that there are alternatives of production, contrary to what the industry discourse wants us to believe. The movement is an attempt to rescue a logic of collaboration so that society thrives with the help of all, a call for all to take responsibility for this prosperity, playing roles of solidarity rather than roles of domination and exploitation.

References

Duménil, G. and Lévy, D., 2003. Superação da crise, ameaças de crises e novo capitalismo. In: F. Chesnais, G. Duménil, D. Lévy and I. Wallerstein, 2003. Uma nova fase do capitalismo? São Paulo: Xamã. pp.15-41.

Hardin, G., 1968. The Tragedy of the Commons. Science, 162(13), pp.1243-1248.

Harvey, D., 2003. The new imperialism. New York: Oxford University Press.

Kelty, C. M., 2008. Two bits: the cultural significance of free software. Durham: Duke University Press.

Marx, K., 2013. O capital: Crítica da economia política. Livro I: O processo de produção do capital. São Paulo: Boitempo.

Ostrom, E., 1990. Governing the Commons: The Evolution of Institutions for Collective Action. Cambridge: Cambridge University Press.

Prado, E. F. S., 2005. Desmedida do valor: Crítica da pós-grande indústria. São Paulo: Xamã.

Stallman, R. M., 2002. Free Software, Free Society: Selected Essays of Richard M. Stallman. Boston: GNU Press.

Torres, A. L., 2018. A tecnoutopia do software livre: uma história do projeto técnico e político do GNU. São Paulo: Alameda.

Williams, S., 2002. Free as in Freedom: Richard Stallman's Crusade for Free Software. California: O'Reilly Media.

---

1 “GNU’s Not Unix!” works as an recursive acronym to represents the idea that the GNU was based in the UNIX operation system, but differs from it, fundamentally, for having an open source-code.

2 Source code is a set of instructions, in a specifically programming language, that composes a computer program. If its code is closed/proprietary, the user cannot access these instructions and is not capable of discovering how a program was built, how it works and, consequently, how can it can be improved, for exemplo.

3 It's important to enforce that “free” in this context does not means free of charge or non-commercial. This confusion arises from the use of the english word “free”, that can mean both. The free software movement does not opposes itself for the commercialization of the software, but to its transformation in a restricted access good through intellectual property mechanisms, such as the copyright.

4 The GPL is a free software licence created in 1989 and maintained until today by the Free Software Foundation, which is also responsible for the maintenance of the GNU Project, the free operation system that started the movement in defense of the open source software.

5 The “copyleft” therm has been used as a pun with copyright, meaning either as “left to copy” as “copy to the left”. It appears to have been created in 1984 or 1985, in a letter written by a friend of Richard Stallman, in which he had written the phrase: “Copyleft - all rights reversed”, in a clear allusion to the phrase that accompanies the copyright notifications: “all rights reserved” (Williams, 2002).

6 There are two main currents within the movement that defends free software, one is called "free software", which is the current that defends the social and political character of the movement; and the other is "open source", which is more concerned with technical and marketing issues of open source. The first one is based on the founder of the movement, Richard Stallman, and the second is Linus Torvalds, creator of Linux, an important component of GNU / Linux operating systems.

7 To delve into this discussion about free software as a "commons" see: Said Vieira, M. What Kind of a Commons Is Free Software? CEUR Workshop Proceedings. Proceedings of the 6th Open Knowledge Conference. Berlin, 2011. Available at: https://ssrn.com/abstract=2619956 [Accessed 20 May 2019].

8 The importance that this model of production has acquired within the industry itself, which sought to abolish it and adopt in its place the closed standard, is indisputable. This can be attested by the massive investment of capital and labor that traditional companies have made in free software projects, as is the case with IBM, which alone will invest $ 1 billion in these projects this year. Source: IBM To Invest $ 1 Billion In Linux! Available at: https://ssrn.com/abstract=2619956 [Accessed 28 February 2019].