What Kind of a Commons is Free Software?

Miguel Said Vieira [⋆]

Faculty of Education, University of São Paulo
São Paulo, Brazil
msaid@usp.br
http://impropriedades.wordpress.com

Abstract. This paper analyzes free software under the light of commons theories, and tries to answer whether it is a managed or open access commons. It briefly presents commons studies and its main concepts, as well as the discussion on immaterial commons, arguing that goods' intrinsic characteristics should not be viewed as absolute, but rather contextualized in social struggles. Then, it proposes a two-tier structure for analyzing free software as a commons, considering its dual nature as source and machine code. The two connected layers of the proposal – use and development – are characterized according to commons theory categories; Android and software forking are explored as examples. It concludes that the first layer resembles an open access commons, but with intensional boundaries, and that the second one resembles multiple managed commons. This disparity is associated with the category of nested enterprises and with the layers' relations to appropriation and production.

Keywords: commons, governance, free software, community

1 Introduction

This paper tries to shed light into an ongoing discussion regarding free software and commons theory: is free software an open access or a managed commons?

While the freedoms granted by free software licenses suggest that it is open access, empirical studies of free software communities also suggest that they are far from anarchical, and that collaboration in those communities follows certain patterns, principles and norms – which obviously suggests that they are managed, rather than open access.

2 Commons

To begin with, it is important to clarify the terms of this debate by presenting a bit of the commons theory upon which it is based. The most important theoretical tradition implied in it is a relatively recent one, based on the work of

[⋆] The author thanks Fapesp (http://fapesp.br) for sponsoring his graduate research, of which this paper is a partial result.

Elinor Ostrom, and which I refer to as the "new institutional" tradition, with reference to Ostrom's declared methodological leaning. One of the greatest boons of Ostrom's work is the fact that it has inspired a wide range of scholars also focusing commons; their work is obviously diverse, and the label "new institutional" should be seen as a mere pointer to this variety of approaches more or less related to Ostrom's.

Her work is a thorough rebuttal of Garrett Hardin's influential claim that a commons was always bound to failure by overuse of its resources [12]. Hardin based this reasoning on an assumption of extreme individual rational self-interest; he suggested that herders in a hypothetical communal grazing field would always abuse the capacity of the common-pool resource, striving for as much of it as possible, and thus leading to its depletion. Ostrom, in turn, argued that, while a model of rational individuals should not be entirely dismissed, other variables had to be factored into this model in order for it to reflect what happens in real-life phenomena of sharing.

She suggested that, when deciding – for instance – whether to (over)use or not a shared resource, individuals weigh in a number of issues beyond immediate self-interest. These issues include: medium- and long-term preoccupations (regarding the sustainability of the resource), the individual's status in the community, the possibility of sanctions, whether or not defections are usual in the community etc. And, through a series of empirical researches on small-scale existing commons (including fisheries, irrigation systems, managed forests), she showed that while some of them do perish, that is not the rule: many others flourish, in some cases since centuries ago.

Her research also pointed out some characteristics (design principles) that can be found in most of those enduring, successful commons, and that are missing in most of the failed ones. Those principles include:

- well defined *boundaries* (both regarding the common-pool resource and the membership in the community);
- *rules* (developed, modifiable and monitored by the community);
- *congruence* between these rules and material conditions;
- and *nested enterprises* to deal with diffent aspects of governance.

Those findings also empirically confirmed a hypothesis that already preceded Ostrom: that open-access commons – those that are open to all – are more prone to depletion; and the reverse with managed commons (those with boundaries and rules). Because of that, she explicitly defines commons as resources shared by a certain community, under certain rules: open access commons are either a formal contradiction, or something doomed to rapidly fulfill Hardin's predicted tragedy.

Returning to the initial question of this paper, this means that determining whether or not free software is managed or open access could also be key in assessing if they are indeed commons, and if so, if they can be expected to be successful and long-enduring.

3 Immaterial commons

However, we should first note that Ostrom focused on a specific type of common-pool resource: local, small-scale commons, based on resources that, in a typology previously developed by economists, are highly *subtractive* or *rival* (that is, those whose appropriation by someone diminishes the amount available for others) and that are to hard to *exclude* others from appropriating. She explicitly took other types of goods, such as public goods, which are non-rival and hard to exclude,[1] out of her analysis.

As has been argued before, in this typology of goods, immaterial goods such as knowledge and information are public goods. When someone copies a book (whatever the means used) from a library, the original does not magically vanish; when someone downloads a copy of a software from the Internet, the previously existing one continues there – unlike what would happen if the resource in question was a bike, for instance, which cannot be simply copied and that can only be used by a single person at a given time.

Sure, a book in a library can only be copied by a single person at a given moment; the copy process could take a long time depending on the means used; and the original could be spent and unusable after many copies (the book as an object is not immaterial, after all). But once a copy is made, others can be made from it, and the cost to produce a copy is usually smaller than that of producing the original. What's more important, once digitization and the Internet join the equation, these "advantages" of immaterial goods are improved by orders of magnitude: now copies can be made simultaneously, very quickly, from long distances, and at a tiny fraction of the cost of a copy of a book (congestion might still be a problem, but a much lesser one).

Even given this difference, Ostrom herself still accepts that in many cases knowledge can still be regarded as a commons:

> Consideration of knowledge as a commons, therefore, suggests that the unifying thread in all commons resources is that they are jointly used, managed by groups of varying sizes and interests. [13, p. 5]

This makes sense when we consider that free software, for example, displays differences in the ways that it is used and managed when compared to other types of immaterial goods, such as proprietary software, which is usually a de facto private good, or a weather forecast (be it produced by the state or by a private organization). In this comparison, free software seems to be much closer to commons as those studied by Ostrom.

3.1 Goods' characteristics are not absolute in a commons

I shall argue that this apparent incongruence is partly due to limitations of the economic typology of goods – or at least due to the belief that this typology

[1] Usual examples of public goods are light from a lighthouse, and public security. The last one is mentioned by Ostrom, who also cites the results of a weather forecast [15, p. 32] as being non-subtractive.

is absolute, and that the goods' economical classification is a sufficient trait to determine how they can be used and shared.

To understand this limitation, one should consider that subtractability and exclusion are not binary, black-and-white variables, but rather a continuum – something which is already acknowledged by economists. And, furthermore, consider that the differences in this variables are not exclusively given, as intrinsic characteristics of the good, but are also influenced by how societies deal with those goods: they are, at least to some extent, also socially constructed.

This can be exemplified in the case of software. Do subtractability and exclusion remain unchanged for a certain piece of software, regardless of the time and place it is being used? TEX, for instance, one of the softwares that was used to generate the file for this paper, has changed very little since it was developed, in 1978; and apart from those changes, in essence this software is still represented in the same way (a series of bytes stored in a medium, be it punched cards or a hard disk) as it was back then: in that sense, its essential characteristics as a piece of information haven't changed much.

However, with the advance in the use of digital technology and the spreading of the Internet, right now it is much easier and cheaper to obtain a copy of this software; as it is also much easier to obtain access to a computer where this copy can be put to use. On the other hand, this is also much easier and cheaper to do that in places such as Europe and the USA; whereas in poorer regions, the resources that are needed to effectively copy this software and put it to use are effectively scarce and rival (including material resources, which are intrinsically more prone to such rivalry). Those resources include the knowledge that is necessary to operate computers and software (which is much more available in the Silicon Valley than it is in probably anywhere in Africa), but also material goods – by definition more prone to subtractability – such as hardware, Internet infrastructure, and the energy and materials that they consume (something which seems less trivial every passing year, even in a rich country).[2]

Additionally, the manifestation of subtractability in a certain part of the world might even depend on its limitation somewhere else: due to the characteristics of Internet infrastructure and governance, poorer countries pay more (in interconnection costs) for Internet bandwidth than richer countries, byte-per-byte; a Kenyan pays a foreign company when he sends an e-mail to a rich country, and he's also the one who pays when he receives e-mails from that country [6].

It'd be very hard (or even outright racist, in the limit) to sustain that these differences correspond exclusively to the "natural" or intrinsic behaviors of information goods in each time and, particularly, in each place. The variables that appear as intrinsic to the goods themselves are in fact, as I have shown, also dependent on social and historical conditions. That does not mean that variables such as subtractability and exclusion do not play any role, but that they

[2] This complicates the distinction between pure immaterial and material commons, as their effective use will always depend on a mix of both types of goods; I argue that the distinction is valid, but not in absolute terms.

should not be taken as objective absolute traits, and rather must be judged and assessed in a wider context, taking into account political human life.

The same reasoning should be applied to material goods: even though they are more inherently scarce and subtractable, it does not follow that every time they are made private this should be seen as "natural" and wholly justifiable on the grounds of their intrinsic characteristics. This privatization should be seen as a result of social struggle and power relations; in many cases, communities can devise ways to make the sharing of such resources possible to some degree, as Ostrom's work exemplifies. Property relations are the result of political interaction, and one should not believe that for every good, an optimal type of property (either private, common, or public / state-owned) can be mechanically determined based exclusively on the good's intrinsic characteristics.

Peter Linebaugh, a Marxist historian that can be placed in a differing tradition of commons thought (with regard to the "new institutional" one), stresses this relationship between the community and the things being shared through the concept of *commoning* [14]. "There's no commons without commoning", as scholar Massimo de Angelis puts it, commenting on Linebaugh: the existence of a commons depends on "life flow", on historically situated human and social struggle that, given a certain minimum level of material conditions, enables that commons [8, p. 9]. While this idea is not totally absent from Ostrom's theory, it is certainly shadowed by the fact that her approach (bearing the mark of much of contemporary mainstream economics) generally follows methodological individualism: that is, she analyses commons and communities first and foremost from the perspective of the individual; historical, social and power relations considerations come in second place.

3.2 An additional peculiarity of software

Now, to approach free software as a commons, one must also take into account the peculiarities of software per se, which go beyond those outlined until now – those of knowledge as an immaterial good.

The most important peculiarity here applies to software in general (not only free software). It is the fact that functioning software exists in two different forms or representations: source code, and machine (or object) code. Source code usually is stored in text files, and is written in languages (programming languages, such as C, Perl and Java) that, while requiring learning, are quite more easily understandable than object code. Source code determines what the software should do and how it should behave, but abstracting from the functioning details of the hardware.

Machine code, in turn, exists in binary form: a sequence of bits that cannot be read by humans, but that computers can understand as a sequence of instructions. Object code is generated through compiling source code; this process translates source code into corresponding machine instructions that (a certain kind of) computers can understand and perform.

This dual nature of software is strictly related to its use and development: it is machine code that is *used*, in the sense that it is the code that is actually executed by computers, thus performing the software's functions; but software is developed originally in source code, and in general can be altered only when source code is available. While it is easy to generate machine code from source code, the reverse is not true. This makes software quite different from many other knowledge-based goods, such as works of literature, where a similar distinction does not exist or is much less drastic.

It also means that, while sharing of machine code can form a very basic type of commons, this sharing reinforces an asymmetry between those who are mere users of the machine code, and the person who controls access to source code: only this person can know everything that the software actually does, and how it does so (something that allows this person to keep certain activities of the software secret, for instance); and only this person can provide improved or corrected versions of the machine code – this is not trivial, as software is generally a very short-lived good if unmaintained (because of bugs, complexity and interdependency). Free software attempts to counter this asymmetry, markedly present in non-free software, by demanding the sharing of source code as well, and thus opens the possibility of blurring the distinction between users and developers of the software.

4 Free software as a commons: Two layers of analysis

Due in large part to this dual nature of software, I propose that when approaching free software as a commons, we should structure our analysis in two layers: one related to use in general of the software, and the other related to its development. These two layers are not completely independent, but each of them will have a specific set of communities, boundaries, rules and governance devices. This two-tiered structure will help in solving the dilemma initially posed in this paper.

4.1 First layer: use in general

The wider of those layers is that of the use of the software in a more general way. It may include occasional practices of software modification, but they are not done in a systematic and structured way (that will only be the focus in the second layer);[3] everyone that uses it, independent of their participation in developing the software, is a member of the *community* of this commons. When we look at this layer, the *governance* of the free software commons is ruled almost exclusively by its licenses and the freedoms (particularly the freedom to use) that they guarantee. The Free Software Foundation's Free Software Definition and the Open Source Initiative's Open Source Definition can be jointly taken as

[3] For an idea of what I mean by occasional development in this layer, contrast a single person changing a certain detail in a free software that he/she uses – and eventually redistributing it, but not presented as a fork of the original software; against, on the second layer, groups of many developers working systematically in a common, more or less structured project.

a lowest common denominator for the principles of these commons. While there are differences[4] and also incompatibility between some of those licenses, in this layer this does not stops us from looking at the whole pool of free software as forming a single commons (that is, the *resource pool boundaries* of this commons include all the pieces of software that are shared), at least with respect to one major feature they they have in common: the mere use of the software is not restricted by any of those licenses.

The commons that can be seen in this layer lies somewhere between an open access and a managed commons, but closer to the former. If we look exclusively at the practice of use, which is the most important in this layer, it is effectively open access.[5] It resembles a managed commons when software is redistributed (a precondition to its sharing), because the commons has rules when this happens; but in general these rules only require that members do not withholding source code, as this would interfere with the freedom of the recipients (and, in the case of non-permissive or copyleft licenses, they require that modified versions of the software remain inside the commons, by following the same license as the original software). Thus, abiding by them is only a matter of will of the member-to-be, and as long as those simple rules are followed, membership can not be refused with basis on ad-hoc rules or on limits to the size of the community – something which might be crucial to the continued existence of material commons.

Based in the vocabulary of logic, I propose that what we have in this layer is an *intensional* definition of the community's boundaries (based on a generic criterion to determine who is a member: compliance to rules on software redistribution), unlike the *extensional* definitions that usually characterize managed commons (based on an enumeration of members, or on ad-hoc criteria such as "relative to family x" or "inhabitant of region y").

4.2 Second layer: development

While free software does open the formal possibility of blurring the distinction between users and developers, this separation is also determined by a variety of factors, from personal interest to social conditions. In practice, the group of people that routinely and systematically develops software – that is, the members of the *communities* in this layer – is a subset of the community of the previous layer.[6]

[4] Particularly in regard to how *permissive* they are; the GPL and the BSD licenses being good examples of non-permissive and permissive licenses, respectively.

[5] In a similar way to what was discussed in topic 3.1, this does not mean that this open access characteristic is absolute: it is constrained by material, social and historical conditions, such as access to Internet, hardware and some technical knowledge; as well as by other characteristics of the software being shared, such as its language and its relevance to local conditions.

[6] And that is true even when we consider, as I do here, *development* as encompassing not only actual coding, but also bug-testing, translating, documenting, evangelizing etc. Since 2010, the Debian project also adopts a similar, expanded notion of "de-

In this layer, however, both *community* and *resource boundaries* are multiplied. Every single community that forms around specific instances or projects of free software counts as a different commons, with its particular (even though in some cases similar) rules, boundaries and systems of governance. In other words, each group of people who not only use a certain free software, but also help to support or develop it (in the expanded sense outlined previously), can be seen as a commons in itself. These commons exist based on both large bundles of software, such as operating systems (for example, most GNU/Linux distributions and BSD-derivatives have associated development communities), or based on particular pieces of software (many of the software projects found in websites such as FreshMeat or SourceForge feature such communities). This proliferation of commons can be explained by the fact that those communities (as the pieces of software they're based on) are diverse, and that they entail – with regard to development, their main activity – drivers, needs and principles that are also very diverse.

The *governance* processes of the many commons that we encounter in this second layer are to some extent also governed by the software license. However, unlike what we see in the first layer, where the license is the single definer of the governance, in the second layer there are many other systems, rules and norms (formal or not) at play, which sometimes are even more prominent than the licenses themselves.

Good examples of formal principles and rules of this kind are the Debian Social Contract and the Debian Constitution, for the community around the Debian GNU/Linux distribution. While the Social Contract represents general fundamental principles that orientate the common goal of the project,[7] the Constitution spells out (sometimes in minute detail) decision-making procedures, hierarchies and prerogatives of the members of the community. Informal rules and principles, on the other hand, are much harder to point out precisely, but they clearly arise routinely in the interactions in those communities, and can affect the majority of decisions inside those commons: from a simple edit in a wiki page, to a code commit that implies defining changes regarding the software, or the acceptance of a significant package in a GNU/Linux distribution. A quick glance at the profuse and heated discussion lists of such projects can attest to the fact that even when they have formal rules are in place, those rules leave room for the development of informal ones.

While in some aspects the membership and effective participation in these communities might seem reasonably open, in most of them it is restricted according to different criteria.[8] Formal systems of governance can imply criteria

veloper" [9], which is also a term reserved for a formal position in that project's structure.

[7] Interestingly, the Social Contract explicitly places the licenses in a subordinated role, as it – the Social Contract – also defines the guidelines for acceptable licenses.

[8] Note that effective participation here does not mean, for instance, merely coding a modification for the software, but actually getting it presented (and considered for a possible commit) inside the project structure.

based on democratic or formal authority principles; frequently, a "meritocratic" criterion (with regard to the dedication of members and the quality of their contributions, for instance) is also implicitly applied [17].[9]

Android as an example The criteria can vary a lot according to the community, its structure, its leaders and sponsors. Google, for example, enforces a tightly controlled governance system for Android – a software stack which has both free and non-free software, with an operating system based on the Linux kernel which is presented as open source:

> Android is intentionally and explicitly an open-source – as opposed to free software – effort: a group of organizations with shared needs has pooled resources to collaborate on a single implementation of a shared product. [3]

However, despite this self-declaration as open source, when the Honeycomb version of Android was shipped in a single flagship device (a tablet, not a regular mobile phone), Google did not publish its source code, and announced that it would be released only "when it is ready" (for phones in general) [16].

Those facts help to characterize a very singular community. While the system is quite open for those who develop applications that run on top of Android [10], it has been described as "completely closed for the handset OEM (pre-load) ecosystem. There is no other platform which is so asymmetrical in terms of its governance structures" [7]. Through this governance system, Google effectively controls what contributions are accepted or not in its official branch; and it does so not only based on meritocratic criteria, but also on its business strategies:

> In the code-review process, an Approver decides whether to include or exclude a change. Project Leads (who are typically employed by Google) choose the Approvers [...]. A Project Lead for an individual project is responsible for the following: [...] Be fair and unbiased while reviewing changes. Accept or reject patches based on technical merit *and alignment with the Android strategy.* [2, emphasis added]

While this closed governance system is allegedly geared to avoiding incompatible implementations, it could also relate to avoiding changes in the platform that would threaten or pose competition to Google's business models. Google has the authority to deny modifications in the system if it considers them incompatible, and it has done so when a software limited the functioning of Google's own positioning system [4]. In an earlier private e-mail to Google's CEO, Andy Rubin – the founder of Android, Inc. and as of 2011 Senior Vice President of Mobile at Google – wrote that mobile location data was "extremely valuable to Google" [5].

Even if the case of Android could be pointed as an extreme example of tight control (and not the norm in free software governance), it is still reasonable to

[9] For an interesting analysis of how the governance system in the Debian project balances meritocracy, democratic participation and formal leadership, see [11].

say that the characteristics of this second layer make it much closer to managed commons than to open access, unlike the first layer.

4.3 Connections between the two layers

Finally, it is important to notice that these two layers of analysis are not independent, even though their outcomes and functioning might even be eventually contradictory. This is quite noticeable in the event of a fork in a free software project – that is, when developers take the code from a free software and start an independent and distinct free software project.

While the possibility of forking a project is a direct and logic consequence of the freedoms granted in a free software license (that is, this possibility is clearly implied as a right in the principles of the first layer), the effective use of this right is not taken lightly in the context of the second layer. As the famous Jargon File puts it:

> Forking is considered a Bad Thing – not merely because it implies a lot of wasted effort in the future, but because forks tend to be accompanied by a great deal of strife and acrimony between the successor groups over issues of legitimacy, succession, and design direction. There is serious social pressure against forking. As a result, major forks [...] are rare [...]. [1]

While the right to fork is determined in a formal and abstract sense in the first layer, the legitimacy of its exercise will be determined in the second layer, on the ground, and subject to the particularities of the case and communities at hand.

5 Conclusion

The first layer of the analysis sketched here paints free software as a single open access commons (although with *intensional* boundaries) when it comes to the general use of the software; the second layer, however, paints it as composed of many managed commons, with varied formal and informal systems of governance and restricted communities.

This incongruence can be read as a complex manifestation of nested enterprises [15] in a single commons, although more comparison to other empirical examples of nested enterprises is necessary to ascertain this. It also appears to mirror the fact that, while the activities in the first layer are centered in the "appropriation" of the resource – the use of the software –, the second layer is focused in the production of such software. Because of that, the first layer strongly benefits from the non-rivalry that characterizes software (even if only in a relative way, as I have argued); in the second layer, in turn, what seems to be pooled or shared is mainly the efforts of the developers (and if software can be non-rival, hours of work of developers definitely cannot).

References

1. forked. In: Raymond, E.S. (ed.) Jargon File 4.4.7. http://www.catb.org/jargon/html/F/forked.html
2. People and roles | android open source, http://source.android.com/source/roles.html
3. Philosophy and goals | android open source, http://source.android.com/about/philosophy.html
4. SKYHOOK WIRELESS, INC. vs. GOOGLE, INC., http://www.socialaw.com/slip.htm?cid=20416&sid=121
5. Your location 'extremely valuable' to google (May 2011), http://www.ibtimes.com/articles/139985/20110501/your-location-extremely-valuable-to-google.htm
6. Bell, R.: The halfway proposition: Background paper on reverse subsidy of g8 countries by african ISPs (2002), http://www.wougnet.org/WSIS/ug/WSIS2005/docs/HalfwayProposition_Draft4.pdf
7. Constantinou, A.: Is android evil? (Apr 2010), http://www.visionmobile.com/blog/2010/04/is-android-evil/
8. De Angelis, M.: Introduction. The Commoner (11), 1 (2006), http://www.commoner.org.uk/?p=24
9. Debian: General resolution: Debian project members (Oct 2010), http://www.debian.org/vote/2010/vote_002
10. Elmer-DeWitt, P.: Why it's harder to make money on android than on apple's iOS, http://tech.fortune.cnn.com/2011/05/27/why-its-harder-to-make-money-on-android-than-on-apples-ios/#
11. Ferraro, F., O'Mahony, S.: The emergence of governance in an open source community. Academy of Management Journal 50(5), 1079–1106
12. Hardin, G.: The tragedy of the commons. Science 162(3859), 12431248 (1968), http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.124.3859&rep=rep1&type=pdf
13. Hess, C., Ostrom, E. (eds.): Understanding Knowledge as a Commons: From Theory to Practice. MIT Press, Cambridge, Mass (2007)
14. Linebaugh, P.: The Magna Carta Manifesto: Liberties and Commons for All. University of California Press, Berkeley (2008)
15. Ostrom, E.: Governing the Commons: The Evolution of Institutions for Collective Action. The Political economy of institutions and decisions, Cambridge University Press, Cambridge (1990)
16. Rubin, A.: Android developers blog: I think im having a gene amdahl moment (http://goo.gl/7v4kf) (Apr 2011), http://android-developers.blogspot.com/2011/04/i-think-im-having-gene-amdahl-moment.html
17. Weber, S.: The Success of Open Source. Harvard University Press, Cambridge, MA (2004)