Open Source Ecosystems and their need for a legal framework

Daniel M. German

August 28, 2018

1 Introduction

The simplest—and most effective—definition of open source software is software that is licensed under an open source license. This definition shifts the definition from the technical domain to the legal one, but more importantly, emphasizes that without open source licenses there would be no open source software. The Open Source Initiative (OSI) defined the characteristics that an open source license should have [9], and has approved 82 licenses as open source[8]. Nonetheless, many other licenses exist that arguably satisfy the requirements defined by OSI (such as the License of Ruby, and the *Do What the Fuck You Want To Public License*).[22, 4].

The ecosystem of open source is very large and it is not a single monolithic entity. This ecosystem is composed, among others, of the many (smaller) ecosystems that are created around each of the different open source software systems. In these smaller ecosystems, their members participate with the expectation of gaining one or more benefits in return. These benefits might be, among others, ethical (the good feeling of helping others), educational (experience gained in the use of certain technologies), or pecuniary (direct or indirect revenue derived from participating in the ecosystem).

The license of an open source system lays the foundation on top of which ecosystems of open source systems are created. It explains how the users can benefit (or not) from development, use and exploitation of the software system. For some, participating in a particular ecosystem might also be an ideological act, which in some cases might be emphasized by the text of the license (for example, the preamble of the General Public License–GPL—addresses political issues beyond intellectual property law). For others, it does not matter how the software is used (the Do What the Fuck You Want To Public License—WTFPL—states that the copyright owners of the system have no interest on what others do with the software, but want to keep its copyright—rather than place the software in the public domain). Netscape, when it open sourced its Navigator, found that no license satisfied its need, as a corporation, for potential commercial exploitation and created its own license, the Netscape Public License. This license, and its derivatives—the different versions of the Mozilla Public License—explicitly define what commercial use is.

The following sections make the argument that licenses and open source ecosystems are symbiotic: one cannot exist without the other. The ecosystem is created around a license; however, as the ecosystem evolves, it creates the need to adapt its license to its new needs of the ecosystem. However, licenses are often not enough, and ecosystems need to complement them with a proper organization (a foundation) whose purpose is to create a framework in which the different entities of the ecosystem can collaborate together to further the success of the software system.

2 Building an ecosystem of users and developers: the need for a license

Open software ecosystems can be traced back to the early days of computing. According to the Wikipedia[29], early users of computers shared software and collaborated in its improvement as if it was in the public domain. Computers were shipped with the source code of their operating systems, allowing users to debug it, customize it, and improve it.

One important moment in the history of open source licensing was the early days of Emacs, the customizable editor, one of the oldest open source software applications still in use today. Emacs was created by Richard Stallman in 1975[16, 17]. As of 1981, the (very informal) license of Emacs read:

> [Emacs] is distributed on a basis of communal sharing, which means that all improvements must be given back to be incorporated and distributed.

The success of Emacs prompted many to port it to other platforms and to create new implementations from scratch. One of these ports was the first implementations of Emacs for Unix, written by James Gosling (creator of Java)[18]. Originally Gosling allowed others to redistribute and modify his implementation. By then an ecosystem of users and developers existed, whose goal was to keep improving Emacs. Stallman argues that a major part of the success of Emacs was the feedback loop between users and developers and that Emacs was the first extensible editor, allowing users to become developers of extensions to the editor [16]. Gosling was an eager member of this Emacs community; according to Stallman [16]: "[Gosling] wrote in a manual that he called the program Emacs hoping that others in the community would improve it until it was worthy of that name".

Stallman reused some of Gosling's code in his own port of Emacs for Unix, GNU Emacs (the first project released by the GNU Project)[27]. However, in 1983 Gosling sold the rights to his code to Unipress. Immediately after Unipress asked Stallman to stop using any code from Gosling's Emacs (Stallman complied). A major outcome of these events was Stallman's realization that, in order to guarantee the spirit of communal work that he envisioned, he had to draft a proper copyright license for Emacs. This license became the GNU Emacs Public License and it evolved to become the GNU General Public License, published in 1988.

3 A software system, its licensing and its ecosystem

It can be argued that any software system creates an ecosystem around it. At its minimum, the ecosystem is composed of the developer (or developers) and its users. Larger ecosystems are composed of many different types of actors. For example, it might include intermediaries (such as the store where the users acquire the software), entities providing training and support, developers who might create plug-ins to improve the features of the software, and many different types of users. In this context, an ecosystem might also include a place for exchange of knowledge or ideas (e.g. a tag in StackOverflow or Quora, or a mailing list), and a market (where services and products are exchanged for some type of consideration).

A user requires a license to be able to use the software system. There might also be a fee associated in the acquisition of such license. This license imposes some rights and obligations, usually related to copyright law. At the minimum it will allow the user to run the program, often with specific constraints. It might restrict its use for certain purposes[1], limit the number of users that can run it, or the number of times it can be run, or limit the period of time when the software can be run. In some cases, the license might even impose limitations on the output of the software, even if the output involves the user. For example, it is not uncommon for licenses to forbid the use of the software to create a competing product (such as Bitkeeper, the version control system used by Linux in the early 2000s did; similarly, the license of Bison, a compiler of compilers developed by the Free Software Foundation, imposes some limitations on its use for the creation of compilers of compilers).

[1]For example, some licenses do not allow the software to be run for benchmarking purposes.

However, a license—particularly an open source license–can also widely expand the rights of how the software can be used that otherwise are forbidden by copyright law. For example, the right to run the software for any purpose, the right to create copies of the software and distribute them for free or for a fee, the right to resell or give away the software, the right to create derivative works, etc.

In essence, the license determines a basic set of rules under which the actors in the ecosystem can participate. For example, an entity interested in providing training for the software must be able to acquire a license that allows it to run the software for such purpose. A developer interested in improving the software requires a license that allows such improvements, either as plugins (this would also require an architecture that supports this feature) or by modifying the underlying software.

Contracts are often the legal mechanism used to grant a license to a user (their most common type is known as End-User Agreement). Another legal mechanism are pure copyright licenses, that do not require a legal agreement between the two parties (such as the license one acquires when one buys an album or a movie). Other contracts that apply to the software might not be related to how users use the software, but how other actors can participate in the ecosystem. For example, a retailer might have a contract that guarantees it to be the exclusive entity that can market and sell the software system.

4 GPL and Free Software: software by the users to the users

The ecosystem that Richard Stallman envisioned in his GNU Manifesto[13] is summarized in the Four Freedoms of Free Software, which are the cornerstone of his movement: the freedom to run the program as you wish, the freedom to study how the program works (access to the source code), the freedom to redistribute copies, and the freedom to modify the program and redistribute copies of modified program. "Roughly it means the users have the freedom to run, copy, distribute, study, change and improve the software." [21]. Free Software is software that guarantees to its users these four freedoms.

Stallman had described how for-profit entities had started to make it difficult to improve and share software [14]. In his view, software is by the users for the users. In [23], he writes: "we designed the GNU General Public License (GNU GPL) to release [the GNU Project software systems] under a license designed specifically to protect freedom for all users of a program". Hence, the GPL lays the legal foundation in which an ecosystem can guarantee that software will be Free Software, and that users will retain their four freedoms. Users (and developers) flocked to this ecosystem and its success is without question.

When Linus Torvalds released his first version of Linux, he was concerned with issues similar to those that had affected Stallman: he was worried that for-profit entities would unfairly benefit from his software without remunerating his work. The first release of Linux (version 0.01, 1991) comes with an ad-hoc license created by Linus requiring that any redistribution (in full or in part) should make all the source code available, copyright notices should be left intact, and no fee can be charged for redistribution[32]. Linus starts the creation of a kernel as an effort of a user, to the users, in a similar manner than Stallman had started the GNU Project before him (Linus benefited tremendously from the GNU Project: he included many of the GNU tools along with his kernel, so users could use his kernel). Linus soon realized the benefits of joining the ecosystem created by the FSF, and in 1992 changed the license of Linux to the GPL-2: "Making Linux GPL'd was definitely the best thing I ever did."[35].

At first sight, Free Software might be intended to alienate for-profit entities. Cygnus Solutions was one of the first participants of this type to join this ecosystem (it merged with RedHat in 2000). According to its founder, Michael Tiemann, the FSF ecosystem created a business opportunity [31]: "companies that provided commercial services (customizations, enhancements, bug fixes, support) based on that [Free] [S]oftware could capitalize on the economies of scale and broad appeal of this new kind of software." Cygnus Solutions took advantage of this opportunity and became, by 1998, the largest open source company in the world [31].

For-profit entities were welcome to join the ecosystem, as long as their participation keeps the Free Software guarantee to its users (i.e. it follows its license). While they are able to sell copies of software they create, others can then copy them many times again without having to pay an extra fee. In other words, they can only expect to profit from the first sell, but the ecosystem welcomes their participation in selling services around free software. Today, most Free Software reaches users through for-profit entities who sell it as part of hardware devices (TVs, routers, embedded devices, all using Android or, Linux) or as distributions of GNU/Linux (RedHat, Ubuntu, Suse). Even Apple's OS X distributes a large collection of Free Software.

5 The academic licenses: do as you wish

An early realization of Stallman is that his employment precluded his ability to create Free Software. In most jurisdictions, in the absence of any other agreement, the employer is the owner of the intellectual property created by the employee. This was one of the reasons that he quit MIT in 1984 [13].

Others were faced with similar challenges: how to create software that could be shared and enhanced by others, without the constraints that intellectual property law imposed? The solution came in the form of academic licenses, that allow the copyright holder (usually an academic institution) to relinquish most intellectual property rights to the software.

One of the oldest of these licenses is the MIT/X11. In the 1980's MIT researchers were working on the creation of a GUI interface for Unix computers called X-Window System (X, for short). The project, called Athena, was jointly sponsored by DEC, IBM and MIT. Both IBM and DEC built Unix computers at the time and were fighting each other for a share of the Unix marketplace; however, they realized that working together to create a GUI for Unix they could increase the size of the market—which, in the long term, would benefit both companies. Athena created an ecosystem in which academics and for-profit participants worked together with a common goal. It is likely that MIT researchers, DEC and IBM wanted to benefit from any software being developed, but at the same time foster an ecosystem in which others (individuals and organizations) would feel welcome to collaborate in its further development. The solution was to draft a license that allowed anybody to do anything they wished with the software—that was otherwise forbidden by copyright law—as long as the copyright notices in the files were not removed. This license became known as MIT/X11, and today is one of the most popular open source licenses.

Researchers at University of California Berkeley had a similar challenge while developing a new version of Unix called the Berkeley Software Distribution (BSD). The academic operating systems community (specially the Unix community) was being encumbered by intellectual property issues that limited their ability to share and enhance the works of others. In 1988 Version 4.3BSD-Tahoe of BSD is released under a license that is likely derived from the MIT/X11. Its main difference is the restriction that the name of the University (the copyright holder) "may not be used to endorse or promote products derived from this software without specific prior written permission."[26]. This license would eventually become the BSD-4, and further derived into the BSD-3 and BSD-2 licenses.

These licenses, the MIT/X11 and the BSD-4, create ecosystems where the software is for anybody to use, for whatever purpose. They welcome for-profit organizations who do not need to feel restricted in the way the participate in the system. These organizations could take the software, further modified it and sell it, without any requirement to share the improvements or profits.

The MIT/X11 and BSD-4 licenses influenced the University of Illinois/NCSA Open Source License (UIUC) that was used in the HTTP server that NCSA (located at University of Illinois) was developing. This license was likely one of the major reasons that this server became popular and—arguably—-partially responsible for the success of the World-Wide Web. Such was the success in the creation of an ecosystem around httpd, that its development was taken over by its users, creating in 1995 the Apache HTTP Server[2], the most successful Web server to date[5]. The creation of Apache HTTP Server was possible because the license of NCSA's http server allowed it.

6 The other IPs: trademarks and patents

As the success of Apache HTTP Server continued, it developers noted the need to protect its name. This is likely among the reasons they changed its license and release the Server under the Apache License version 1.0 (and later, version 1.1). The main difference between the adapted BSD license (from which it is derived) is the addition of the following clause that protect the name of the software system (present in both versions of the license):

> Products derived from this software may not be called "Apache" nor
> may "Apache" appear in their names without prior written permission
> of the Apache Group.

It appears that one of the advantages of the permissiveness of the original Apache License was also its disadvantage: it allowed members of the ecosystem to commercialize the software system, even under the Apache name, creating confusion among other participants of the ecosystem. Avoiding this type of confusion is one of the major goals of trademarks.[3]

IBM, as a leader in the field, was also starting to experiment with this nascent method to develop and distribute software. However, IBM did not embrace any of the licenses at the time and decided to create its own, the IBM Public License (IPL). The main differences of this license with respect to its predecessors are: placing liability not on the creator of the software but on the distributor, and the addition of clauses that specifically address patents. The software that used this license had little impact, but the IPL became the predecessor of the Common Public License (CPL) that later evolved into the Eclipse Public License (EPL).

[2]It is often mentioned that its name is short for "A PATCHy sErver" in reference to its origins as a set of patches on top of the NCSA server.

[3]This need to protect the name of Apache was one of the major motivators for the creation of the Apache Software Foundation [1].

7 Open source: the need to create a larger ecosystem

The proliferation of licenses for these open projects was creating barriers for collaboration between their corresponding ecosystems, even though they had a similar goals in mind: to collaborate in the development of software where the source code could be openly shared, enhanced, and redistributed (with or without enhancements). Furthermore, many of these licenses were considered incompatible with each other (e.g. the original BSD-4 license and the all GPL versions are incompatible) restricting potential sharing of software between the different ecosystems[2]. Nonetheless, new ecosystems were being created around these software projects (and their ecosystems), such as Linux distributions—most notably Debian. Debian's goal has been to create a Free Software operating system. One of the major challenges that Debian faced was the evaluation of different licenses to determine if they were Free Software (this process is documented in the Debian Free Software Guidelines[20]). Debian is one of the first to try to harmonize different software products under the umbrella of Free Software. By doing this, it enlarges the ecosystem by integrating Free Software developed under difference licenses,

In 1999, the Open Source Initiative (OSI) is created with two major goals in mind [7]. The first is to define a new term, Open Source, that formalizes many of the goals of the communities around these open collaboration systems; this goal is accomplished with the Open Source Definition, a set of 10 requirements that a software license must have to be called Open Source (these requirements are derived from the Debian Free Software Guidelines). The second is the creation of a certification process by which licenses can be approved to be called *Open Source*. The term Open Source was also created to "distinguish it from the philosophically- and politically-focused label 'Free Software.'" [7] and to incorporate under its umbrella licenses that were not considered to be Free Software, but that satisfied the basic requirements of open collaboration (such as the BSD-4 or the EPL). The term Open Source unifies all the communities behind each of the projects that use any of the OSI-approved licenses under a larger ecosystem: the Open Source community.

The success of Open Source prompted the interest of for-profit organizations to participate in the Open Source ecosystems—and to benefit from them. License compliance has become a necessity for these organizations; they want to make sure that, if they use Open Source, they do it in the correct way—first and foremost—by satisfying the requirements that the licenses of the software they use. License compliance has become an important concern for those using Open Source (specially when the goal is to make profit). This has lead to the creation of for-profit organizations that assist others in license compliance (Blackduck is today the leader in this area) and community efforts to create resources to educate developer and organizations on best-practices. For example, the Linux Foundation has been sponsoring efforts towards license compliance, such as the Software Package Data Exchange format—SPDX, whose goal is to standardize how licensing information should be shared between organizations using Open Source software [12] and the Open Compliance Program [25], whose goal is to educate and help developers and companies to properly comply with license requirements.

Other organizations were created by authors of Open Source software with the explicit goal of guaranteeing that those using the software comply with its license. The most important being GPL Violations, whose goal is to raise awareness regarding infringement of GPL'ed software, and to assist developers in the enforcement of their copyrights.[34].

8 The Foundations: the need to go beyond the license

In 1985, the Free Software Foundation is formed to create a legal structure around its software [28]. It holds the copyright of a major portion of the software developed by the FSF, accepts donations to continue its development and is the steward of the GPL family of licenses.

Apache HTTP Server became so successful at creating an ecosystem around it, that it required a formalized way to organize its users. In 1999 the Apache Foundation is established. Four of its most important reasons for existence are: [1]: to provide hardware and business support to the community of developers, to serve as a legal entity that can receive donations, to shield members from potential legal risks, and to protect the Apache trademark.

Over the years many other foundations have been created. Some by individuals for the individuals (e.g. GNOME Foundation, the Software Freedom Conservancy), others by for-profit companies interested in creating an independent entity to anchor the growing ecosystem (e.g. KDE e.V., the Eclipse Foundation), and finally, as a consortium of for-profit organizations to collaboratively foster the ecosystem's development (e.g. The Linux Foundation, the OpenStack Foundation).

While the bylaws of each of these organizations differ, they all have one feature in common: they become the center of activity around the software system (or systems), and by extension, the center of their corresponding ecosystems.

Some foundations also create a strong set of rules that determine how other software systems join the ecosystem. For example, both, the Apache Foundation and the Open Stack Foundation have strong rules in terms of how a new software system can become part of its ecosystem. This includes what license is should have (e.g. Apache Foundation only accepts software systems that use the Apache License or a license compatible with it [19]).

9 License evolution: The need to adapt to the environment

As time passes by, the license used by a software system becomes inadequate to address the needs of the its ecosystem, specially those of its main contributors and users. For example, the GNU GCC compiler, one of the first Free Software projects, generates binaries that need an implementation of the C standard library to run. Without this library, C programs are not very useful. The GCC project faced two alternatives: write its own C Standard library, or generate code that could have been linked to somebody else's C Standard library. The second alternative as impractical: it would have eroded the freedoms of its users, since it would have made GNU GCC dependent on acquiring a license to such C Standard library (there was no Free Software implementation at the time). The first alternative meant that this library would have been licensed under the GPL too. If the license of the library was the GPL, then any binary compiled against it would have to be licensed under the GPL too. Effectively, GNU GCC would only be usable to build Free Software. This issue raised interesting ethical concerns: *should Free software be only used to create Free Software only?* or, *should the person who runs the software have the* freedom *to create binaries under a difference license?* It became obvious that for GNU GCC to succeed (and it did succeed) users should be allowed to do anything they wanted with the binaries they created. The ecosystem for GNU GCC needed to grow beyond the restrictions imposed by the GPL. According to Richard Stallman, "using the GPL for [the GCC C Standard library] would have driven proprietary software developers to use another [compiler]".[15]

The solution was the creation of the Library General Public License (today known as the Lesser General Public License–LGPL). The LGPL is primarily designed to be used by a library, and it allows a program that links to such library to have any license. The LPGL tries to achieve a balance between the goals of Free Software (the library cannot be modified without making these modifications Free Software) and the needs of proprietary software (to create software that they can license—and sell—as its owner pleases.

There are many other instances in which licenses have evolved due to the needs of its users. One of the major motivations for the creation of the GPL-3 was the need to address patents. The Apache License 1.1 removes the main advertising clause, present in the version 1.0, something that made it difficult to commercialize products based upon it. The IBM Public License evolved into the Common Public License because of the need to remove IBM from the license text, hence making the license usable by others.

As mentioned before, licenses, when incompatible between each other, foster isolation of their ecosystems. The original BSD-4 license is incompatible with the GPL family of licenses. Software developed under this license cannot be combined into derivative works under the GPL license. The Mozilla Public License version 1 and the Eclipse Public License are also incompatible with the GPL. Even worse is that some licenses are incompatible with licenses in their same family. Software licensed under the GPL-2 is incompatible with software licensed under the GPL-3.[4]

However, if the goal of Open Source is to create software that can be shared, enhanced, and further distributed, license incompatibilities have become a barrier between communities, sometimes in ironic ways: any organization can enhance and further distribute code under the BSD-4 and license the results under a restrictive license, yet a software project licensed under the GPL cannot do it. License compatibility has been one of the major reasons licenses have been updated. For example, the Mozilla Public License 2.0 added clauses that explicitly address compatibility with the GPL family of licenses.

10 Fairness: rules on how to collaborate

Any ecosystem is expected to have internal struggles between its participants, and open source ecosystems are not an exception. This struggle is, perhaps, one of the major contributors to its natural evolution and adaption—without it, it might simply cease to exist.

The FSF has been, since its conception, concerned with keeping the entire copyright of its most important software systems. This guarantees the FSF the ability to relicence the software in any way its sees fit (usually, by changing the license to a new version of the GPL). Contributors to FSF's software are requested to transfer their copyright of their improvements; otherwise these improvements will not be incorporated. This position has lead to split of ecosystems. When some Emacs developers were not willing (or capable, since their copyrights were sometimes owned by their employers) they created XEmacs, dividing the Emacs community and its ecosystem[30]. See [6] for a study of the motivations of developers to fork, which includes licensing issues.

In other ecosystems the struggles have been around control. For example, the Gnome Project was very concerned that for-profit organizations could hijack the project from its users. The main body responsible for taking decisions is the Board

[4]This is the reason why the Free Software Foundation recommends that software be licensed under *the General Public License version 3 or any further version*; this way, when a new version of the license is published, software can be relicensed under the new version, making it unnecessary for the new version to be compatible with the older one.

of Directors of the Gnome Foundation. The bylaws of the Gnome Foundation state that no “organization, corporation or similar entity, or any affiliate of shall hold more than 40% of the Board [of Directors] seats.” [24] Other ecosystems, primarily those created by for-profit organizations, see it differently and give more power to specific entities. For example, the Open Stack Foundation categorizes members into three types: individuals, Platinum (further divided into Class A and B), and Gold. Platinum members, which have to pay higher fees than Gold members, received certain privileges in return (each Platinum member can appoint a member foe the Board of Directors, while Gold members are represented by persons who are voted whom they vote for).[10]

The difference between the structure and organization of the Board of Directors of these foundations highlights the typical struggles of distributive fairness found in any collaborative environment. Distributive fairness is concerned with how resources should be allocated to each member of a system (see [3] for a description of fairness within the scope of software engineering). On one hand, a system can strive for equality: each member should have the same benefits as any other member, regardless of its amount of participation (as Gnome does, where any contributor can be a member of the foundation and be elected to its Board of Directors–the Gnome Foundation bylaws define the process and minimum requirements to achieve membership). On the other hand, a system can strive for equity: each member receives a benefit from the system that is proportional to its participation (as Open Stack does, where does who contribute more—Platinum members, who pay a membership of 500k US$, compared to Gold members who pay at most 200k US$[11]—receive more benefits from the system, in this case their ability to control the project’s direction). It is not clear if one system is better than the other, but it certainly affects the decision making process of their corresponding ecosystems.

Another important type of fairness is interactional fairness. Interactional fairness is concerned with the interpersonal treatment of people in a system. It reflects the degree in which people are treated with politeness, dignity and respect. While merit has been seen as the main factor that determined membership and status in an Open Source ecosystem, it became clear that it was necessary to create rules that defined how people were to treat others. As described by Tourani et al, each ecosystem addresses this issue in a different way: “the phrasing of a code of conduct, enforcement mechanisms used, scope and other properties vary depending on the code of conduct and [its] community”. [33].

The rise of the foundations around open source systems, the decision processes and the code of conduct rules that these projects select demonstrates that the licenses—a prerequisite of an Open Source ecosystem—are not necessarily sufficient to guarantee the successful collaboration of its members.

11 Conclusion

The license of an Open Source system becomes the legal framework that outlines the rights and responsibilities of the ecosystem participants. Without an Open Source license, and Open Source system cannot exist, nor its ecosystem. With the pass of time, the needs of the main stakeholders of the ecosystem evolve, which might force a chance in its licence.

The creation of a foundation is a major landmark in the evolution of an ecosystem. It legally formalizes the relationship between the different members of the ecosystem, and its decision process. The bylaws of a foundation (a legal document itself) complement the license of its system. A foundation is capable of changing the license of the system (and often changes it) in response to the needs of its members.

Projects have also complemented the rules of participation with rules of conduct. These rules might not be legally enforceable, but give the ecosystem the ability to expel a non-compliant member.

The nature of any social ecosystem requires rules that outline its members participate, collaborate and benefit from it. Ecosystems have flourished around open source systems because their licenses have created the social contract that guarantees that the needs of its of its members are satisfied. And when licenses have stop doing this, the ecosystems find ways to adjust, because, after all, an ecosystem's goal is to continue flourishing.

References

[1] Apache Software Foundation. Frequently asked questions. `http://apache.org/foundation/faq.html`, 2017.

[2] D. M. German and A. E. Hassan. License integration patterns: Addressing license mismatches in component-based development. In *31st International Conference on Software Engineering, ICSE 2009, May 16-24, 2009, Vancouver, Canada, Proceedings*, pages 188–198, 2009.

[3] D. M. German, G. Robles, G. Poo-Caamaño, X. Yang, H. Iida, and K. Inoue. "was my contribution fairly reviewed?": a framework to study the perception of fairness in modern code reviews. In *Proceedings of the 40th International Conference on Software Engineering, ICSE 2018, Gothenburg, Sweden, May 27 - June 03, 2018*, pages 523–534, 2018.

[4] R. M. Meloca, G. Pinto, L. Baiser, M. Mattos, I. Polato, I. S. Wiese, and D. M. German. Understanding the usage, impact, and adoption of non-osi

approved licenses. In *Proceedings of the 15th International Conference on Mining Software Repositories, MSR 2018, Gothenburg, Sweden, May 28-29, 2018*, pages 270–280, 2018.

[5] Netcraft Inc. July 2017 Web Server Survey. https://news.netcraft.com/archives/2017/07/20/july-2017-web-server-survey.html, July 2017.

[6] L. Nyman and T. Mikkonen. To fork or not to fork: Fork motivations in sourceforge projects. In S. A. Hissam, B. Russo, M. G. de Mendonça Neto, and F. Kon, editors, *Open Source Systems: Grounding Research: 7th IFIP WG 2.13 International Conference, OSS 2011, Salvador, Brazil, October 6-7, 2011. Proceedings*, pages 259–268, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[7] Open Source Initiative. History of the OSI. https://opensource.org/history, 1998.

[8] Open Source Initiative. Licenses by name. https://opensource.org/licenses/alphabetical, 1998.

[9] Open Source Initiative. The Open Source Definition. https://opensource.org/osd, 1998.

[10] Open Stack Foundation. Bylaws of the OpenStack Foundation. https://www.openstack.org/legal/bylaws-of-the-openstack-foundation/, 2014.

[11] Open Stack Foundation. Proposed Budget and Funding Structure. https://wiki.openstack.org/wiki/Governance/Foundation/Funding, 2017.

[12] SPDX Working Group. About The Software Package Data Exchange (SPDX. https://spdx.org/, 2017.

[13] R. Stallman. The GNU Manifesto. Dr. Dobb's Journal of Software Tools, http://www.drdobbs.com/open-source/the-gnu-manifesto/222200498, Sept. 1985.

[14] R. Stallman. *Open Sources: Voices from the Open Source Revolution*, chapter The GNU Operating System and the Free Software Movement. O'Reilly, 1st Edition edition, Jan 1999.

[15] R. Stallman. Why you shouldn't use the Lesser GPL for your next library. https://www.gnu.org/licenses/why-not-lgpl.en.html, 2017.

[16] R. M. Stallman. Emacs: The extensible, customizable, self-documenting display editor. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1979.

[17] R. M. Stallman. Emacs the extensible, customizable self-documenting display editor. In *Proceedings of the ACM SIGPLAN SIGOA Symposium on Text Manipulation*, pages 147–156, New York, NY, USA, 1981. ACM.

[18] L.-C. A. Ta. The History of the GPL. https://www.free-soft.org/gpl_history/, July 2001.

[19] The Apache Software Foundation. Incubation Policy. http://incubator.apache.org/incubation/Incubation_Policy.html, 2017.

[20] The Debian Project. Debian Social Contract version 1.0. https://www.debian.org/social_contract.1.0, July 1997.

[21] The Free Software Foundation. What is free software? https://www.gnu.org/philosophy/free-sw.en.html, 2016.

[22] The Free Software Foundation. Various Licenses and Comments about Them. https://www.gnu.org/licenses/license-list.en.html, 2017.

[23] The Free Software Foundation. Why Open Source misses the point of Free Software. https://www.gnu.org/philosophy/open-source-misses-the-point.en.html, 2017.

[24] The GNOME Foundation. Bylaws of GNOME Foundation as of April 5, 2002. https://www.gnome.org/wp-content/uploads/2012/02/bylaws.pdf, Apr 2002.

[25] The Linux Foundation. About the Open Compliance Program. https://compliance.linuxfoundation.org/about-open-compliance-program, 2017.

[26] The Wikipedia Foundation. Berkeley software distribution. https://en.wikipedia.org/wiki/Berkeley_Software_Distribution, 2017.

[27] The Wikipedia Foundation. Emacs. https://en.wikipedia.org/wiki/Emacs, 2017.

[28] The Wikipedia Foundation. Free Software Foundation. https://en.wikipedia.org/wiki/Free_Software_Foundation, 2017.

[29] The Wikipedia Foundation. History of free and open-source software. https://en.wikipedia.org/wiki/History_of_free_and_open-source_software, 2017.

[30] The Wikipedia Foundation. Xemcas. https://en.wikipedia.org/wiki/XEmacs, 2017.

[31] M. Tiemann. *Open Sources: Voices from the Open Source Revolution*, chapter Future of Cygnus Solutions—An Entrepreneur's Account. O'Reilly, 1st Edition edition, Jan 1999.

[32] L. Torvalds. Linux version 0.01. http://ftp.funet.fi/pub/linux/historical/kernel/old-versions/RELNOTES-0.01, Aug 1991.

[33] P. Tourani, B. Adams, and A. Serebrenik. Code of conduct in open source projects. *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 00:24–33, 2017.

[34] A. Vance. The defenders of free software. The New York Times, http://www.nytimes.com/2010/09/26/business/26ping.html, Sept 25 2010.

[35] H. Yamagata. The Pragmatist of Free Software: Linus Torvalds Interview. Tokyo Linux Users Group, 1997.