Harvard Journal of Law & Technology
Volume 30, Special Symposium

OPPORTUNISTIC FREE AND OPEN SOURCE SOFTWARE DEVELOPMENT PATHWAYS

*Greg R. Vetter**

TABLE OF CONTENTS

I. INTRODUCTION

Every software license limits opportunism in some sense. This holds for proprietary licenses as well as for licenses this Article will call "free and open source software" ("FOSS") licenses. The license terms reflect the opportunism to be prevented. Sometimes the terms amplify the rights the license deploys. Thus, a proprietary software

* HIPLA Professor of Law, University of Houston Law Center (UHLC); Co-Director, Institute for Intellectual Property and Information Law (IPIL). For helpful comments and discussion, I thank: Participants at the Private Law and Intellectual Property Symposium, held in March 2016, by the Project on the Foundations of Private Law at Harvard Law School; Jorge Contreras; Christina Mulligan; Dave Fagundes; and Sapna Kumar.

license prohibits copying even though the software is covered by copyright's reproduction right.[1]

Sometimes the terms of a software license invert its rights-base. For example, a FOSS license allows copying and distribution for any type of use, but requires attribution to the originator. The opportunism impeded is taking credit for the work of another.[2] Copyright's orientation is prohibiting unauthorized reproductions, but the FOSS license allows copying in furtherance of a goal: impeding some variety of opportunism in software development and information technology. FOSS licenses reflect the dissatisfaction some communities express with intellectual property protection in software. To these communities, a better alternative is no property rights in software, but failing that, software under FOSS licenses is preferable to software under proprietary licenses.

Given that some licenses work against intellectual property in software, conflicting opportunism-impedance strategies among different licenses[3] reflect tension about property rights in software. This Article's claim is that Henry Smith's modularity framework for intellectual property rights[4] gives greater insight into this tension. Inhibiting opportunism with use of a resource is part of modularity, so this Article uses a definition of opportunism inspired from Smith's work with platforms and equity: opportunism is undesirable behavior, in part because the actions are contrary to the purpose of the property

---

1. DOUGLAS E. PHILLIPS, THE SOFTWARE LICENSE UNVEILED 10 (2009). Of course, under the Copyright Act's preemption clause, 17 U.S.C. § 301, the contractual prohibition might not be effective. *See* MICHAEL RUSTAD, SOFTWARE LICENSING § 5.12[5], at 489–91 (2015) (discussing the copyright preemption test for license terms).

2. Greg R. Vetter, *The Collaborative Integrity of Open-Source Software*, 2004 UTAH L. REV. 563, 611 (2004) [hereinafter Vetter, *Collaborative Integrity*].

3. The opportunism strategies relate to differing conceptions about software modes of production, competition within the information technology ecology, and how to monetize software technology. Greg R. Vetter, *Commercial Free and Open Source Software — Knowledge Production, Hybrid Appropriability & Patents*, 77 FORDHAM L. REV 2087, 2096–100, 2109–14 (2009) [hereinafter Vetter, *Commercial FOSS*]. This Article is steeped in discussion of software development, information technology markets, and FOSS licensing. Relying on my background expertise in these areas and my prior scholarship, I will cite minimally for these topics. *See, e.g.*, Greg R. Vetter, *Slouching Toward Open Innovation: Free and Open Source Software (FOSS) for Electronic Health Information*, 30 WASH. U. J.L. & POL'Y 179 (2009) (discussing FOSS in the enterprise software market). For a practical overview of FOSS licensing, see HEATHER J. MEEKER, THE OPEN SOURCE ALTERNATIVE: UNDERSTANDING RISKS AND LEVERAGING OPPORTUNITIES (2008). For an anthropological treatment, see CHRISTOPHER M. KELTY, TWOBITS: THE CULTURAL SIGNIFICANCE OF FREE SOFTWARE (2008), http://twobits.net [https://perma.cc/C9KK-RGDQ].

4. Henry E. Smith, *Intellectual Property as Property: Delineating Entitlements in Information*, 117 YALE L.J. 1742, 1779–82 (2007) [hereinafter Smith, *IP as Property*]. *See also* Henry E. Smith, *Property and Property Rules*, 79 N.Y.U. L. REV. 1719, 1774–85 (2004) [hereinafter Smith, *Property Rules*] (discussing how an exclusion strategy for property rights has advantages for avoiding opportunism, particularly as opportunism arises from asymmetric information).

rights;[5] it "is residual behavior that would be contracted away if ex ante transaction costs were lower."[6] Even with this definition, opportunism is a difficult concept to cabin. It is relative, relational, and depends on past positions among parties.

Licenses are not the same as intellectual property rights in software. Licenses shape what users may do with the software. Ubiquitous licenses can transcend the public/private divide to attain a quasi-public character.[7] Thus, license rights may be more important for users than the underlying intellectual property rights.[8]

Smith's modularity framework is based on information costs shaping the scope of opportunism-impeding intellectual property rights, and is an extension of his information costs approach to real property rights.[9] In a real property context, ex ante, trespass law allows the owner to engage in a variety of uses according to her valuation of those uses or other preferences. The power to exclude associated with the trespass rule gives the owner an incentive to develop information about possible uses and associated values and costs.[10] The right to exclude is the informational signal that acts as a boundary between the modules. One module of human activity is the

---

5. Henry E. Smith, *Property as Platform: Coordinating Standards for Technological Innovation*, 9 J. COMPETITION L. & ECON. 1057, 1062, 1078 (2013) [hereinafter Smith, *Property as Platform*].

6. Henry E. Smith, An Economic Analysis of Law Versus Equity 9–11 (Mar. 27, 2012) (unpublished manuscript), http://www.law.uchicago.edu/files/files/Smith%20paper.pdf [https://perma.cc/868F-W3WD] (noting that "the most salient feature of opportunism is the difficulty in defining it"). Part of Smith's definition of opportunism for purposes of his equity discussion is purposefully not included: "behavior that is undesirable but that cannot be cost-effectively captured — defined, detected, and deterred — by explicit ex ante rulemaking." *Id.* This is omitted because property rules and licenses are ex ante, but licenses in particular are subject to opportunism in ways that property rules are not. For example, a licensing party might decide to breach the license based on asymmetric information or other considerations. Finally, this Article uses "strategic behavior" and "opportunism" synonymously.

7. NANCY S. KIM, WRAP CONTRACTS: FOUNDATIONS AND RAMIFICATIONS 59–62 (2013); MARGARET JANE RADIN, BOILERPLATE: THE FINE PRINT, VANISHING RIGHTS, AND THE RULE OF LAW 33–40 (2013). Licenses make a private instrument, with particular opportunism impeding goals, an instrument of competition policy from an interest group. Indeed, some licenses evangelize software production modes, seeking to originate and grow software under that license, use of the license in other software, or both. Paradoxically, a license's deployment of private rights may transform a popular license into a quasi-public instrument. This effect is the specific goal of many FOSS licenses.

8. The rights base for software licenses include the four traditional subareas of intellectual property: trade secrecy, copyright, patent, and trademark. Among these four, this Article addresses only license concepts for trade secrecy and copyright because these two work in concert in proprietary licenses but against each other in some FOSS licenses.

9. Smith, *Property as Platform*, *supra* note 6, at 1059, 1078; Smith, *IP as Property*, *supra* note 4, at 1742–43, 1765–66.

10. The exclusion strategy is in contrast to a liability rules approach, where information production by all possible parties is under a different profile of incentives in light of an adjudicator valuation at the end of a process where the trespasser has a right of entry subject to damages. Smith, *Property Rules*, *supra* note 4, at 1727–31.

owner(s) and what they do with the land. The module on the other side of the interface is all possible trespassers. Another modularity example in real property is the law of waste acting as an interface between the module of future interest owners as compared to the module of life estate owners.[11]

To explain modularity generally, this Article transforms real property examples into a spatial metaphor: modularity seeks to locate the property rights interface so that it clusters most human interactions around a resource on either side of a boundary made by the rights. The shape and character of the rights define the modules of interactions partitioned by the rights interface.[12] The trick is to find the sweet spot for the interface such that the interface costs[13] are less than the benefits of diminished opportunism with the recourse covered by the rights. This spot is where most interactions with the resource occur within a module on either side of the interface. The interactions within each module do not have the opportunism opposed by the interface.

This Article posits that it is difficult for software to achieve a stable modularity interface to impede opportunism.[14] In other words, it is difficult, compared to other types of information resources, to prevent strategic behavior both ex ante and ex post. This is due to the multimodal complexity of software development, the overlapping rights-bases applicable to software, and the licenses that underlie distribution and use of software.[15] In particular, this Article's focus is the contrast and incompatibility among FOSS license types against the

11. Smith, *Property as Platform*, *supra* note 6, at 1061.

12. The interface might be primarily an exclusion strategy, or a governance strategy of use rights, or a hybrid of each. Smith, *IP as Property*, *supra* note 4, at 1749–50, 1765, 1785, 1799.

13. Interface costs include the information costs needed to administer and delineate the rights of the interface, whether that interface is based on an exclusion strategy or a governance strategy. Smith, *IP as Property*, *supra* note 4, at 1801 (noting that "[p]atent and copyright differ in many ways, but especially in the costliness of delineating and evaluating use").

14. This Article is the first application to FOSS of Smith's modularity framework for delineating entitlements in information. I have previously written about opportunism in the context of software standards and FOSS. Greg R. Vetter, *Open Source Licensing & Scattering Opportunism in Software Standards*, 48 B.C. L. REV. 225 (2007) [hereinafter Vetter, *Scattering*]. There has been prior commentary about FOSS license incompatibility and the related problem of license proliferation. *See, e.g.*, Robert W. Gomulkiewicz, *Open Source License Proliferation: Helpful Diversity or Hopeless Confusion*, 30 WASH. U. J.L. & POL'Y 261, 263−64 (2009). Some license incompatibility is by design, such as the GPL, with terms intended to unsettle the proprietary licensing approach.

15. In Smith's treatment of modularity for intellectual property rights, he compares it to the law of accession. Smith, *IP as Property*, *supra* note 4, at 1766–77. Whatever the complexity of accession for chattels, the complexity of software combinations, and clearing and disentangling rights for use of software in systems, seems to dwarf the information cost problems of accession.

background of the proprietary software license.[16] I use the term "pathways" to describe software that is relicensed over time as different developer groups work with the software: sometimes a relicensing attempt will succeed because the earlier license is compatible with the terms of the later license; sometimes the relicensing attempt produces incompatibility, rendering the later license ineffective.

Success for a FOSS license includes use of the software it covers, or its application to more software, or both. While there are many FOSS licenses, a taxonomy of two general types is important. First, permissive licenses allow any downstream use or relicensing of the software, including into proprietary licensed software, so long as certain attributions are given.

The second type is what this Article calls copyleft licenses. Emblematic of copyleft licenses is the General Public License ("GPL").[17] Copyleft licenses demand greater continuance of the terms of the license, which typically includes that the software's source code remain non-secret and that free distribution not be encumbered (by, for example, ongoing royalty payment obligations for distribution recipients).[18] Generally, copyleft licenses have greater complexity and potential effect on the technological future for software covered by the license. The development pathway of the software — the manner in which software can be developed — depends on the FOSS license initially selected.

Success for a FOSS license also includes impeding some variety of opportunism. License terms indicate behavior deemed strategic in ways incompatible with a desired production modality for the software.[19] The choice of license for a FOSS project, and the potential

---

16. The incompatibilities that arise with FOSS licenses are a particularly poignant observation hinting that full modularity is difficult to achieve for software because certain FOSS license creators intended to alter the modularity interface, that is, disavow or alter intellectual property rights in software. They have achieved significant success in doing so: there is a voluminous amount of FOSS that underlies much public infrastructure and private information technology.

17. There are two important versions of the GPL. Free Software Found., *GNU General Public License, version 2*, GNU OPERATING SYS. (Apr. 12, 2014, 12:39 PM), http://www.gnu.org/licenses/old-licenses/gpl-2.0.html [https://perma.cc/292Z-CWKL] [hereinafter GPLv2]. GPLv2 did not explicitly handle granting and terminating permissions to practice software patent rights. This, along with the need for various other changes, resulted in version 3 of the GPL. *See* Free Software Found., *GNU General Public License*, GNU OPERATING SYS. (Nov. 8, 2014, 3:04 PM), http://www.gnu.org/licenses/gpl-3.0.html [https://perma.cc/UK7L-YDBH] [hereinafter GPLv3]; *Rationale for First Discussion Draft*, FREE SOFTWARE FOUND., http://gplv3.fsf.org/gpl-rationale-2006-01-16.html [https://perma.cc/F9UX-BAUD] (discussing the decision to create version 3 of the GPL).

18. Greg R. Vetter, *Exit and Voice in Free and Open Source Software Licensing: Moderating the Rein over Software Users*, 85 OR. L. REV. 183, 202 (2006).

19. Jay P. Kesan & Rajiv C. Shah, *Deconstructing Code*, 6 YALE J.L. & TECH. 277, 362–66, 2003–04.

influence of the original license on the software development process both shape the opportunism impeding possibilities.

From a modularity perspective, where opportunism dampening helps situate a rights interface between modules of interactions around the resource, the pros and cons of license-pathway potentials influence the calculus in selecting a FOSS license. Choosing a license is tantamount to locating the interface, or relocating it in relation to the rights-base.

Additionally, license complexity may increase interface costs such that future pathways for use of the technology are constrained or require transaction costs to obtain. In such a situation, the foregone uses might represent opportunism dampening that does not overcome the cost of the license-rights-interface or the lost opportunity of the foregone uses, despite the other uses allowed by the license.[20]

To demonstrate these principles with typical FOSS licenses, Part II provides a brief overview of permissive versus copyleft FOSS licenses. Next, Part III models the rights-base of copyright within the modularity framework, overlaying software licenses. The insight is that the use of software under different licenses, over time, might create pathway incompatibilities. These incompatibilities suggest that a stable interface for the modularity framework is more difficult to achieve for software compared to other types of information resources covered by intellectual property rights. Then, Parts IV and V, respectively, discuss these principles from the perspective of two software scenarios. Finally, Part VI concludes.

II. FREE AND OPEN SOURCE SOFTWARE ("FOSS") LICENSING APPROACHES

The hallmark of FOSS development is available source code. In contrast, the proprietary method of licensing software keeps the source code as a trade secret and licenses the executable software. But while FOSS generally is associated with available source code, one of the two general FOSS license types, the permissive license, does not, within the license, require available source code. The second approach, the copyleft license, conditions activity with the software on the source code remaining published and available.

---

20. An example of a license-rights-interface with high information costs and foregone uses is the GPL. By requiring that source code not be incorporated into proprietary licensed software, the GPL imposes a cost of foregone uses, or imposes the cost of a transaction to attempt to relicense the source code from the contributors. Additionally, particularity for GPLv3, understanding the license requires one to overcome significant complexity. The cost to obtain that understanding is itself an information cost that influences the efficacy of the GPL as a license-rights-interface.

A characteristic feature of software development is the interacting nature of software. Code layers on other code, entangled by lesser or greater degrees of mixing and interfacing.[21] The reach of copyright in code, particularly with respect to the derivative work right, extends the licensing possibilities. Comingled source code from different FOSS licenses might not be compatible in a legal sense even if the software will interwork.[22] License compatibility significantly affects the productive uses possible for FOSS code.[23]

A. The Permissive License Approach

The permissive license approach allows one to copy, modify and distribute the software for any type of downstream use. The primary condition is to give attribution to the originator. The permissiveness is broad: the FOSS project can be taken into a proprietary software application where distribution does not include source code.

The permissive license raises several puzzles about how FOSS projects under this approach succeed. The license does not require source code availability, so a variety of influences might keep the source code available. These are described elsewhere,[24] so these puzzles are put aside beyond the observation that a permissive license does not impede the opportunism that these other influences seek to cabin.

B. The Copyleft Approach

A copyleft license conditions activity with the software on the source code remaining available. As used in this Article, copyleft is synonymous with the General Public License. The GPL introduced

---

21. *See* Vetter, *Collaborative Integrity*, *supra* note 2, at 635–36, 674 n.361.

22. MEEKER, *supra* note 3, at 59–62.

23. *See* Gomulkiewicz, *supra* note 14, at 281–82. FOSS licenses are form licenses published on the Internet. There are literally hundreds, with many possible taxonomies. But to simplify the discussion, this Article will use the typical dichotomy: permissive licenses in contrast to copyleft licenses. The essential rights base for FOSS licenses is copyright. With copyright, a license might condition redistribution of the software on an action by the recipient. For example, to redistribute, one must give attribution to the source. FOSS license conditions could theoretically be based on any of copyright's applicable rights, but the distribution right is prominently used. For copyleft licenses, however, the derivative work right is additionally important. David McGowan, *Legal Implications of Open-Source Software*, 2001 U. ILL. L. REV. 241, 268, 253–63 (2001).

24. Influences keeping source code available under permissive projects include, in several categories: reputational interests; economic interests, particularly for suppliers of complements; organizational structures exhibiting governance over the code; felt belief in the value of available source code; and a smattering of other influences. STEVEN WEBER, THE SUCCESS OF OPEN SOURCE 46–49, 184–87 (2004); Jonathan M. Barnett, *The Host's Dilemma: Strategic Forfeiture in Platform Markets for Informational Goods*, 124 HARV. L. REV. 1861, 1899–908 (2011).

the copyleft approach into software licenses. The term is a play on words: copyright licenses often narrow and restrict uses with the software and legally enforce the non-availability of source code; "copyleft" licenses do much the opposite.[25]

The GPL imposes conditions on the redistribution of the software. A recipient can use it with minimal conditions, but redistribution with modifications triggers the GPL's requirements to make sure source code is available and without ongoing royalty obligations.[26]

The GPL defines modifications broadly. Intermingling the copyleft software with other code might trigger a condition that the newly intermingled code also be redistributed under the terms of the GPL. This condition is based on copyright's derivative work right. If that newly intermingled code came from software with a proprietary license, the GPL's proposition is that the source code must be disclosed from that proprietary software.[27]

Software under a copyleft license is legally incompatible with insertion into software deployed under a proprietary or permissive license: the permission set of allowed actions with a proprietary or permissive license includes actions disallowed by the copyleft license. Such an insertion produces a pathway incompatibility.

The opposite approach is allowed and sometimes referred to as one-way compatibility: inserting permissively licensed software into copyleft or proprietary software.[28] The copyleft insertion might be achieved by obtaining permission from all the copyright holders in the copyleft software, but this is tantamount to altering its license scheme.

FOSS projects may have small numbers of locatable contributors such that one can obtain either a new license permission or perhaps even a copyright assignment to a single person or entity. But some prominent copyleft software has thousands of contributors,[29] posing a significant hurtle to unifying the copyright ownership in the software or changing the license.

There is another reason copyleft software with many contributors is difficult to relicense. The intermixing of those many contributions into an operable program makes pulling it apart sometimes

---

25. Free Software Found., *What is Copyleft?*, GNU OPERATING SYS. (Oct. 3, 2015, 5:25 PM), http://www.gnu.org/copyleft/copyleft.html [https://perma.cc/SVC6-WKXU].

26. GPLv3, *supra* note 17, at §§ 6, 10.

27. Greg R. Vetter, *"Infectious" Open Source Software: Spreading Incentives or Promoting Resistance?*, 36 RUTGERS L.J. 53, 58, 88–94 (2004) [hereinafter Vetter, *Infectious Open Source*].

28. *Creative Commons BY-SA 4.0 Declared One-Way Compatible with GNU GPL Version 3*, FREE SOFTWARE FOUND. (Oct. 9, 2015, 5:06 PM), https://www.fsf.org/blogs/licensing/creative-commons-by-sa-4-0-declared-one-way-compatible-with-gnu-gpl-version-3 [https://perma.cc/RW4B-BC7Z].

29. Linux Kernel Development, LINUX FOUND. (Feb. 18, 2015), http://www.linuxfoundation.org/news-media/announcements/2015/02/linux-foundation-releases-linux-development-report [https://perma.cc/2AJS-DN92].

impractical. The copyleft software is often a single copyrighted work in executable form. Its use, value, and utility are in that form as a whole. The components of the whole are internally entangled. Those components arose from a diffuse group of programmers who all keep copyright in their contributions but licensed each component to the copyleft software.[30]

C. Forking a Software Development Pathway

When a FOSS project splits into two development groups that take a previously unified body of code down two separate pathways, this is a fork.[31] A fork can occur for software under either license type. Another type of fork is when permissively licensed FOSS is forked to a new development group which continues with the permissive license, but is also relicensed under copyleft to a separate development group.

The fork is a potential pathway alternative for most FOSS licenses. The reasons and purposes of a fork are mostly extralegal, but important in the context of opportunism. Over time, the forked software might exhibit dissimilarities. Alternatively, the two paths of the fork might remain similar in the code but under control by different programming groups.

Consider a fork example based on GPL software. One of the GPL's purposes is to impede the opportunism of secret source code. Another purpose is to ensure that the code base is always available for anyone to fork. For example, a GPL licensed project might attract the attention of a company. Before that, there were a dozen developers who contributed occasionally in a volunteer mode. The company has complementary products and it assigns four full-time programmers to work on the GPL software. The more the company improves the GPL software, the better its business with the complements. The original dozen volunteer programmers dislike this corporate influence, so they fork the code and continue with their own version.

The diagram below illustrates GPL code that forked twice. Two of the versions, A and B, fell into disuse, but version C continued with viability and active developer support.

30. LAWRENCE ROSEN, OPEN SOURCE LICENSING 252–53 (2005).

31. Gregorio Robles & Jesús M. González-Barahona, *A Comprehensive Study of Software Forks: Dates, Reasons and Outcomes*, *in* OPEN SOURCE SYSTEMS: LONG-TERM SUSTAINABILITY 1, 2–3 (Imed Hammouda et al. eds., 2012).

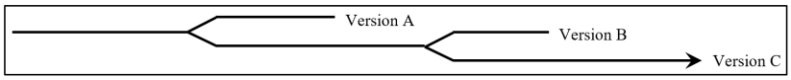

Figure 1: Forked GPL Project Pathway
(Solid line is GPL licensed code)

The figure above uses a GPL example because projects are likely to remain with the GPL once started under that license, particularity when there are many contributors. The license type and the possibilities of forking are both part of the opportunism considerations with FOSS licensing.

The next Part shifts to an explanation of the modularity framework for property and the role within it for opportunism impedance. Linked to those ideas is the idea that software licenses also aim to impede varieties of opportunism.

III. FOSS LICENSES AND THE MODULARITY FRAMEWORK FOR INTELLECTUAL PROPERTY

A. License Interaction, Opportunism, and the Modularity Framework

License incompatibility suggests some discord with Henry Smith's modularity framework when applied to property rights in software. This discord reflects tension some communities in information technology express concerning intellectual property rights in software. The overarching incompatibilities are: the FOSS approach versus the proprietary approach; and the permissive versus copyleft approach.[32]

Modularity seeks to situate an interface within human activity by shaping rights that make up the interface. The legal terms of the rights-interface determine how different classes of users interact among themselves in relation to the resource. Each class of users is a module on one side of the interface. The shaping purpose is to diminish opportunism by partitioning human interactions with the productive resource, driving interactions to either side of the interface created by the rights-base. With software, this is done by a license. But software is an unusual resource in this process. It has the key

32. Different licenses impede different varieties of opportunism. As a result, there are more intricate incompatibility possibilities when examining license terms in greater detail. This Article uses a high level of comparison by contrasting the proprietary license with the two typical FOSS categories. Companies or persons choose one license or another for a variety of reasons, including the emphasis of this Article, to inhibit opportunism in others. That opportunism is in the eye of the beholder, which is why there is a stratification of licenses.

characteristics of other information goods, but the additional characteristic of interoperability in its common productive uses.

Software entangles with other software to interoperate. An outward perspective of this suggests considering network effects, interoperability, standards, and impact on complements within information technology.[33] An inward perspective shows how bundling software into an operable whole may put components into a single derivative work that may or may not have compatible licenses.[34] Both perspectives illuminate the difficulty of arranging a stable modularity interface for rights in software. Both perspectives have their current static impacts, but have dynamic possibilities for change.[35] The dynamic nature of software as a resource compounds the difficulty of shaping a rights-base for it.[36] Licenses might fill that gap, but licenses issue from multiple sources with varying agendas.

The figure below illustrates the difficulty conceived above from the perspective of modularity while also illustrating Smith's modularity framework. The concept of the figure is volumetric and spatial. The rights-base situates an interface with use of the software. The modularity framework envisions the boundary interface to apply such that it is no more costly than the losses from strategic behavior. The interface seeks to locate on either of its sides a maximum volume of human interactions with the software resource. The horizontal axis of the "bowtie" shape is, at the center, a place where a minimum number of transactions occur on the edge of the law, that is, the edge of legality from the perspective of the rights-base. Moving either to the left or right from the center of the bowtie shape signifies moving into a greater volume of human interactions with the resource that steer clear, increasingly, from any conflict with the rights-base.

The thesis is that the interface's location lacks full stability with software as a resource given its current rights-bases and license types. The figure displays licenses in a symbolic way as influences on the interface or departures from it.

---

33. Vetter, *Scattering*, *supra* note 14, at 234–35; Vetter, *Commercial FOSS*, *supra* note 3, at 2118–23.

34. HEATHER MEEKER, OPEN SOURCE FOR BUSINESS 51–56 (2015).

35. Jonathan M. Barnett, *The Illusion of the Commons*, 25 BERKELEY TECH. L.J. 1751, 1775, 1806–08 (2010).

36. Brian Fitzgerald, *Has Open Source Software a Future?*, *in* PERSPECTIVES ON FREE AND OPEN SOURCE SOFTWARE 93–100 (Joseph Feller et al. eds., 2005).

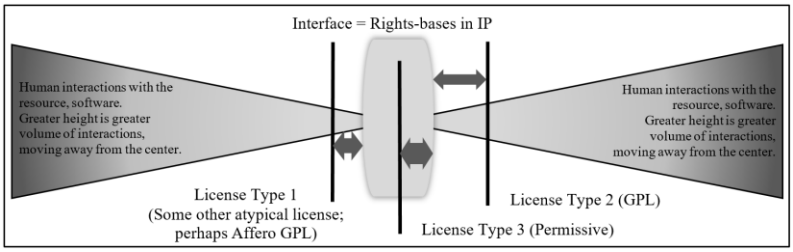

Figure 2: Human Interaction Modules Separated by Interfaces from Rights-Bases and/or Licenses

Figure 2 envisions an interface of rights-bases in software from intellectual property, including trade secrecy, copyright, and patent law. The gray rectangular shape near the middle of the "bowtie" represents the interface. Each end of the bowtie represents modules of human activity on either side of the interface.[37] The bundle of intellectual property rights comprising the rights-base has width along the horizontal axis to signify the range in scope of what the rights allow, and the uncertainly associated with that scope inherent in the rights-base. Licenses sometimes reduce that uncertainty, but sometimes increase it.

The vertical license lines represent licenses with opportunism-impeding goals that seek to shift the interface location. The licenses do not have the full legal power of the rights-bases, but have a quasi-public-law effect. Note that license types one and two tend to unbalance the bowtie modules: less than a maximum number of transactions are partitioned to either side. Assume that license type two is the GPL, one might view the GPL's effect this way: against a world of proprietary licenses, the GPL's effect is to disturb the proprietary scheme; use of GPL software regularly creates incompatibilities when intermixed with other software types.[38]

License type three might be the permissive license: it is a nearly frictionless instrument, so it reduces the complexity of the rights base

37. Ex ante, a purpose of the rights-base is for classes of users to interact among themselves with the resource. A class on one side of the interface modeled in Figure 2 is thought to have greater volumes of interactions where the parties have steered clear of the rights interface. Thus, the figure envisions the horizontal dimension as the degree to which an activity with the resource comes close to violating the rights-base or terms of a license. This activity is near the center of the bowtie diagram. Activity moving away from the center is more clearly a transaction with the resource that has little legal ambiguity and is thus a human interaction that is fully modularized.

38. For the communities that brought forth the GPL and curated its evolution over the last several decades, the goal was to unsettle proprietary licensing and intellectual property right in software. Ronald J. Mann, *Commercializing Open-Source Software: Do Property Rights Still Matter?*, 20 HARV. J.L. & TECH. 1, 27–30 (2006).

itself.[39] License type one is some other atypical FOSS license, of which there are many.

If the licenses and their vagaries make the interface for software shifty, what does this mean? It may mean that the concept of opportunism diminishment in modularity is hard to satisfy with software as a resource. The right-bases covering software are used to create licensing implementations with incompatibilities in important quantities of software in use.[40]

Perhaps software is an information resource where it is difficult to find a stable modularity interface. This seems particularly possible given that copyright's conception of the derivative work right is highly amorphous with software.[41] The FOSS posture disfavoring trade secrecy in software source code also suggests this possibility.[42] The incompatibilities owe their potency in part to copyleft's use of the derivative work right in the rights-base of copyright, and use of copyright itself to eliminate secret source code.

An inward perspective of software interoperability shows how a single copyrighted work might have components with incompatible licenses. An example would be where copyleft software has been injected into a work that is published under a permissive license. This is one of several scenarios covered in the next section.

B. Pathways of Approach and Impeding Opportunism

The discussion earlier in this Part posits that software is a particularly difficult resource for the modularity framework to locate a stable interface. Software has enjoyed several decades where the rights-bases of copyright, trade secret, and patent law underlie licenses targeting opportunism or seeking to support business

---

39. Another example of simplifying a license is the way in which the GPLv2 was applied to the Linux kernel. In order to reduce the uncertainty of the reach of copyright's derivative work right in software, the Linux kernel application of the GPL disclaimed the license terms from reaching any software running on the kernel so long as the application made normal system calls to the kernel as the means of interfacing with it. Vetter, *Infectious Open Source*, *supra* note 27, at 113–18.

40. I recognize in this Article that other explanations might also contribute to an understanding of the problems of license incompatibility. I do not intend the observation, if correct, of insufficient modularity to be the sole explanation. For example, as Christina Mulligan helpfully points out, anticommons effects may also be at work. *See* Christina Mulligan, *A Numerus Clausus Principle for Intellectual Property*, 80 TENN. L. REV. 235, 255–56 (2013) (discussing the possibility for an anticommons effect when multiple copyrighted works are incorporated into a new derivative work).

41. Lateef Mtima, *So Dark the Con(Tu) of Man: The Quest for a Software Derivative Work Right in Section 117*, 69 U. PITT. L. REV. 23, 25–26, 49–50 (2007).

42. Trade secrecy is inconsistent with a software development methodology, like that in FOSS, which emphasizes the public availability of source code. *See* Clark D. Asay, *A Case for the Public Domain*, 74 OHIO ST. L.J. 753, 783–84 (2013); Greg R. Vetter, *A Public Domain Approach to Free and Open Source Software?*, 75 OHIO ST. L.J. FURTHERMORE 8, 10, 17 (2014).

models.[43] Yet there is significant discord and heterogeneity with the licenses.

For example, it is error for a programmer to insert copyleft software into other software under a permissive license. If someone takes the permissive software private without understanding that some of the software is actually licensed under copyleft, the copyright holders in the original copyleft software have an infringement action. A potential partial remedy is to release the source code for the copyleft software. But, if the privatizing entity has intimately entangled its own proprietary code with the copyleft software (falsely posing as permissively licensed software) and distributed the resulting bundle, it may be required to make its own original proprietary source code available. This is a pervasive concern among proprietary software companies.[44]

In the figures below, this section will show some related scenarios where license incompatibility influences pathways for software development.

Figure 3, below, represents a regular occurrence. A proprietary software company arranges to purchase a company to obtain its proprietary software assets. The acquirer, however, discovers that some of the software assets contain copyleft software.[45]

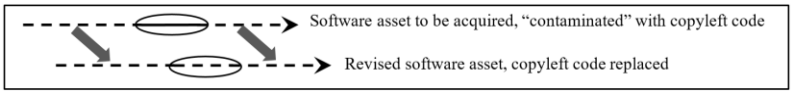

Figure 3: Purging GPL Software for Acquisition
(Dashed line is proprietary licensed code)

The pathway of purging GPL software for an acquisition derives from intellectual property due diligence. The acquirer insists on forensic analysis[46] that discovers the problem: copyleft software incorrectly placed inside a proprietary product. The surprisingly persistent existence of this problem stems from one of the explicit goals of the GPL license: engendering publicly available source code. The GPL seeks to inhibit the opportunism of ongoing royalty payments associated with proprietary licensing. The problem occurs

43. RUSTAD, *supra* note 1, at § 1.06[1].

44. Vetter, *Infectious Open Source*, *supra* note 27, at 152–56.

45. MEEKER, *supra* note 3, at 54–62. Often, the software asset seller's management is not aware of the "contaminated" code, but forensic analysis proves the point. If the problem is pervasive, the buyer might scuttle the sale. If the severity is manageable, the acquisition might go forward with restructuring: the seller is charged with the cost to remove the copyleft software and replace it with software compatible with proprietary licensing.

46. *Open Source Software Audits*, BLACKDUCK, https://www.blackducksoftware.com/on-demand/open-source-software-audits [https://perma.cc/C2E4-58JS].

because the seller's original due diligence to vet code used in its product is insufficient. There is much excellent software freely available on the Internet. The seller's programmers may have helped themselves to these resources without notice to management.

Figure 4's scenario is straightforward, but often hard to realize for copyleft software with many contributors: someone wants to obtain the rights to take the copyleft software private. If successful, the software continues as a proprietary product, as shown in the diagram below.

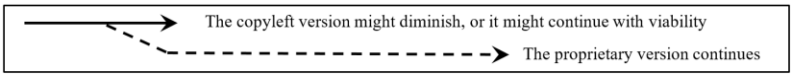

Figure 4: Buyout-Sellout GPL Project Pathway

Copyleft software with many contributors faces various collective action problems among the programmers. Many of those problems relate to management of the technological direction of the software. Typically, the licensing structure is established at the beginning of the project. Indeed, the type of FOSS license used influences who contributes. As a copyleft venture progresses with more and more contributors, it becomes harder to relicense the whole project. But sometimes an external entity can achieve relicensing with financial rewards and a narrative about how it will influence the software positively in the future.

The pathway in Figure 4 is titled "buyout" for the scenario where all contributors holding a copyright are convinced to assign their copyrights to the buyer, or grant a license allowing the buyer to take the software down a proprietary path. The pathway also has the word "sellout" in the title because some programmers might feel that selling the copyright to a privatizing entity is selling out the values associated with copyleft software, and in particular, the GPL license. Any single programmer could hold out. But if the number who hold out is small, their code could be purged in a process that is equivalent to the pathway in Figure 3.

The pathway scenarios covered in this section hinge on the incompatibility that results in certain arrangements of permissive, copyleft, or proprietary software. As software develops over time, it may generate complexities that frustrate the ability of law or licensing to prevent strategic behavior.

Three decades ago the dominant software licensing approach was the proprietary license. The GPL ushered in the copyleft approach with a particular agenda to inhibit certain types of privatizing opportunism with GPL software. All licenses target some sort of opportunism. License selection at the start of a FOSS project involves

difficult estimates of the future.[47] Perhaps the incompatibility issues arise from copyright as the underlying rights-base for FOSS licenses, in particular its amorphous derivative work right. These observations support the claim that software has an uncertain nature in the modularity framework.[48]

The next Part will expand the treatment of path dependence for license selection with two scenarios. Each represents common choices facing FOSS software developers for a new project. As such, the pathways represent opportunism to be avoided or exploited.

IV. FLEXIBLE SOFTWARE PATHWAYS

These pathways posit a FOSS project leader choosing a permissive license for future flexibility. Figure 5, below, shows some possible pathways.

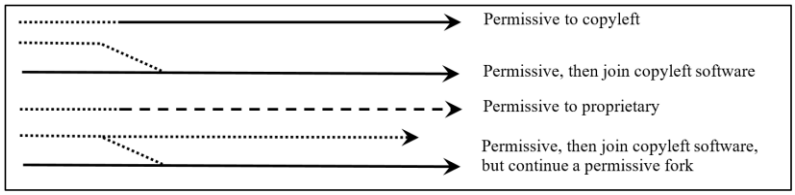

Figure 5: Flexible Software Pathways
(Dotted line is permissive licensed code)

By starting with a permissive license, the leader has a greater range of alternatives. This, however, might necessitate non-license governance structures if the project remains under the permissive license long-term. The structures' purpose would be to facilitate available source code in an ongoing, sustainable way.

During startup, the permissive license lets the leader test the environment. Two possible alternatives are discussed in the sections below. In the first, the project's pathway needs the support of a company. In the second, a copyleft community supports the project. The permissive license's flexibility makes it palatable for a variety of opportunities.

---

47. Molly Shaffer Van Houweling, *The New Servitudes*, 96 GEO. L.J. 885, 900–03, 937–94 (2008).

48. A smaller claim available to the logic of this Article is that copyleft induces the uncertainty for software in the modularity framework for intellectual property rights, but that the uncertainty applies less to other types of software. One reason to adopt the smaller claim even if the larger claim is not persuasive is that copyleft purposefully creates the potential for extreme license incompatibility because it prohibits certain preexisting, popular monetization modalities for software.

A. Fiscal Benefit Potential Under the Permissive License Approach

The straightforward fiscal benefit pathway is to convert the permissively licensed FOSS project into a proprietary software project if there is a paying market.[49] This result might have been intended as a planned strategy. Or, the project might have been public tinkering noticed by a company. Converting the software to a proprietary mode allows direct monetization.

In contrast, converting the project to the GPL renders most monetization opportunities indirect in some way. The complementary goods or services that might coalesce around the software will depend on technological and market factors.[50] The monetization opportunities for some GPL projects are also influenced by the developers' strong sense of ideology for the mode of development.

B. Contributor Recruitment Under the GPL Approach

A FOSS leader starting a new project might encounter a niche where programmer recruitment works best with the GPL approach. Thus, the permissive-to-copyleft pathway allows the leader to make that transition. Once under the GPL, if the software grows with many contributors, there is, in practical effect, no going back.

The pathway ending in copyleft emphasizes that FOSS is also a movement. More specifically, each of the two primary FOSS license types correlate to sub-movements within FOSS. The first is the free software movement, associated with copyleft and the GPL. The second is the open source movement, associated more loosely with the permissive license.

Covering all the differences between the sub-movements goes beyond what is useful to discuss in this Article. But it is worth saying that the free software movement is more directly fueled by ideology.[51] To the extent programmers are willing to voluntarily contribute to a free software project, many will do so only if the software is under a

---

49. One commentator postulates that this scenario is a possible strategy for Apple after it released a programming language under a permissive license. Florian Mueller, *Apple May Regret Its Choice of a Permissive Open Source License for the Swift Programming Language*, FOSS PATENTS (June 9, 2015), http://www.fosspatents.com/2015/06/apple-may-regret-its-choice-of.html [https://perma.cc/5WAC-65TE] ("Apple . . . could always release a future version of Swift . . . exclusively under a proprietary software license. It can't re-close the source code published by then, but it has no obligation to publish more code on open source terms."). *See also* Robles & González-Barahona, *supra* note 31, at 6 (noting that some forks to a FOSS project are due to a commercial strategy).

50. MARTIN FINK, THE BUSINESS AND ECONOMICS OF LINUX AND OPEN SOURCE 175–89 (2003).

51. *See* Richard M. Stallman, *Why Software Should Not Have Owners*, FREE SOFTWARE FREE SOCIETY: SELECTED ESSAYS OF RICHARD M. STALLMAN 45–49 (Joshua Gay ed., 2002).

copyleft license such as the GPL. As a result, FOSS project initiators might find themselves in a situation where they embrace the GPL license even if they are not themselves free software adherents. This is done to make the project more attractive to contributing developers. The instinct might be that this effect is particularly important for developers volunteering their own time, but it might also be important for developers being paid to work on FOSS projects by their primary employer.

This Part emphasized the flexibility of the permissive license when starting a FOSS project. Some sources report recent growth in software under the permissive license in comparison to copyleft.[52] The flexibility of the permissive license might be a causal factor in that trend. Another factor might be information costs comparing the two licenses. The cost to generate information about uses of a resource is an influence within the modularity framework.[53] The more costly the rights-interface, in terms of information costs, the less efficacious it is to impede opportunism. In Figure 2's "bowtie" representation, the permissive license vertical line (License Type 3) is placed near the center because it is an almost frictionless instrument: interactions among humans relying on it have a straightforward character. The GPL copyleft license, on the other hand, pushes away from the center, purposefully outside the character of the rights-base making up the modularity interface because it seeks to upend proprietary software licensing.

The GPL is a complex license compared to the permissive license. Arguably, it is more complex than the copyright rights-base it uses. GPL version 2 originated in 1991, but its complexity increased substantially with version 3 in the mid-2000s.[54] The additional complexity creates information costs that may not be overshadowed by the benefits of inhibiting the varieties of opportunism the GPL targets.[55] The balancing of these concerns plays out in the dynamic and constantly changing ecology that is modern, networked, and

52. Jim Farmer, *Open Source Software Licensing Trends*, OSSWATCH (Feb. 5, 2015), https://osswatch.jiscinvolve.org/wp/2015/02/05/open-source-software-licensing-trends [https://perma.cc/DC3N-LZB6] (discussing trends toward greater use of permissive licensing in comparison to copyleft licensing).

53. Smith, *IP as Property*, *supra* note 4, at 1745.

54. Robert W. Gomulkiewicz, *A First Look at General Public License 3.0*, 24 COMPUTER & INTERNET LAW. 15, 20 (2007).

55. One point of evidence about the increased complexity of GPLv3 is that it lags behind its predecessor in license popularity. *Top 20 Open Source Licenses*, BLACKDUCK, https://www.blackducksoftware.com/top-20-open-source-licenses [https://perma.cc/4DLA-88BH] (showing GPLv2 at 21% while GPLv3 is at 9%); Heather Meeker, *Who's Afraid of GPL3?*, BLACKDUCK (Jan. 25, 2013), http://osdelivers.blackducksoftware.com/2013/01/25/whos-afraid-of-gpl3 [https://perma.cc/RD7V-B3YD]. From an information cost perspective in the modularity framework, the shift from GPLv2 to GPLv3 needs to cause increased opportunism-impeding benefits that overcome the increased information costs of version three of the license.

increasingly personal information technology. To some extent, ubiquitous software licenses are the modularity interface as much as, or more than, the rights-bases used by the licenses. Therefore, if the GPL is being used less, the interface may be shifting or warping.

The next Part addresses software aggregations. It is an apt venue to discuss hybrid approaches potentially combining license types in software systems.

V. AGGREGATION SOFTWARE PATHWAYS

A. Software Aggregations and Hybrid Approaches

Software license incompatibility exists within a copyrighted work. Software aggregations do not intermix all of the software into a single work. For example, the FOSS operating system Linux is actually an aggregation of many FOSS software projects into an operable system.[56] The kernel of the operating system carries the name Linux, but many other dozens or hundreds of FOSS projects are present in a typical distribution of Linux.

A Linux distribution carries the various software components under various licenses. When the operating system runs, some of these components might be properly thought to be combined as a single copyrighted work, but many are not. The Linux kernel itself, while under GPL version 2, explicitly modifies its license to declare that software running on the kernel is not reached by the derivative work right potential so long as it accesses the kernel through normal system calls.[57] For example, one can operate the proprietary Oracle database software on a computer running the Linux kernel without fear that the Oracle software will need to come under the GPL license terms.

Software aggregations containing FOSS can be functioning software systems because the potential license incompatibilities are partitioned. In an aggregation, software components exchange data and cooperate technologically in ways that do not (it is thought) trigger the reach of copyright's derivative work right in software. This is the case with a Linux-based operating system distribution. It is the case with many other software systems that are large and sufficiently complex such that the costs of arranging this technological partitioning is worth it to gain access to the FOSS functionality within the system.[58] The entire system can be distributed with some of the software under a proprietary license and other software under a

56. FINK, *supra* note 50, at 19–21.
57. Vetter, *Infectious Open Source*, *supra* note 27, at 113–18.
58. ROSEN, *supra* note 30, at 253.

copyleft license. The mere colocation of both types on distribution media, or in an electronic distribution stream, does not trigger the GPL's viral effect.[59]

These observations complicate and extend the possibilities for hybrid systems. They show the importance of copyright's construct of a work, taken to its limit by the derivative work right. A base illustration of the pathway possibilities is given below.

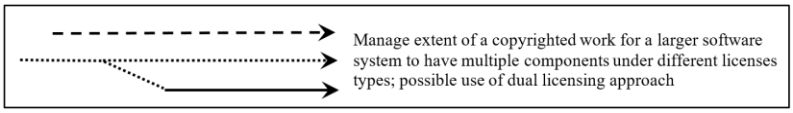

Figure 6: Aggregation Software Pathways

In terms of sequence, the permissive license might originate software projects that take different forms in a system. The stylization in the figure is overly simple. The system is often patched together from FOSS pieces collected across the Internet rather than originating from a few FOSS sources and fragmenting over time.

Recently another consideration has arisen for enterprise software applications: delivering functionality to the user from what is called the cloud. The next section considers this evolution and a FOSS license that aspires to induce transparency for source code in the cloud.

B. FOSS and Source Code in the Cloud

The Internet converted classical concepts of remote computing into "the cloud." Pervasive Internet accessibility enables our resources and data to come from elsewhere. The companies operating cloud services such as social networks and search engines, use much FOSS internally.[60] The GPL licensing structure originated when most software was distributed to run on computers near the user. As a result, the copyleft license terms are triggered by certain types of physical distribution of the software.[61] But software running in the cloud does not amount to a distribution.[62]

The result is that GPL code operating in the cloud, but delivering functionality via an interface to users on the Internet, mostly does not constitute a copyright distribution of software source code by the

59. Vetter, *Infectious Open Source*, *supra* note 27, at 95–97.

60. Jose Teixeira, *Open-Source Technologies Realizing Social Networks: A Multiple Descriptive Case-Study*, *in* OPEN SOURCE SYSTEMS: LONG-TERM SUSTAINABILITY 250, 250–53 (Imed Hammouda et al. eds., 2012).

61. MEEKER, *supra* note 34, at 69–75.

62. MEEKER, *supra* note 34, at 71.

cloud operator.[63] For example, assume that a company takes a publicly posted GPL document management system and sets up a cloud service with it. It attracts a sufficiently large user base that allows it to make a profit on Internet advertising. It regularly improves the code, intimately intermixing its revisions with the original software within a copyright work. The user interactions with the interface do not trigger an obligation under the GPL to provide the improvements. For copyleft adherents, this is a bad result because new software that improves a GPL project is not made available to the community.

One particular group promulgated a license with hopes to impede the opportunism of hiding source code in the cloud.[64] Using the hypothetical document management system, if the original project was licensed under the Affero GPL, then the company's revisions would need to be made available as source code. The Affero GPL uses copyright as it would map to "users interacting with [the program] remotely"[65] — presumably invoking copyright's display right and/or performance right. That remote activity is written as a license condition that requires intimately intermixed revisions to the source code to be made available. Software aggregations and software in the cloud compound the opportunism varieties inherent in the development pathways described in the earlier Parts. In Figure 2's "bowtie" representation, the vertical line for License Type 1 might represent the Affero GPL because it seeks to substantially unsettle the licensing approaches that went before it.

VI. CONCLUSION

Rights in software evolved over many decades, first embracing trade secrecy, then copyright, and later patent law. These rights are deployed by licenses. Thus, licenses provide the full shape and extent of interactions among all of us as users of software. Some licenses, particularly copyleft licenses, seek to shift the modes of production and usage for software in furtherance of a social movement about freedom to access software. Licenses are contested and sometimes incompatible, showing the dynamism of both technological and legal evolution in this important technological field. Beyond their inconvenience, software license incompatibilities suggest a theoretical

63. Whether there is a distribution of source code from the cloud will be technology dependent. Some Internet technologies will send source code from the cloud to the user computer's client application. An example would be copyrighted client-side code such as JavaScript.

64. Free Software Found., *GNU Affero General Public License*, GNU OPERATING SYS. (Nov. 8, 2014, 3:04 PM), http://www.gnu.org/licenses/agpl-3.0.html [https://perma.cc/6FQA-MYE9].

65. *Id*. at § 13.

observation: from the perspective of Henry Smith's modularity framework for intellectual property rights, software may have an uncertain rights-interface within that framework.