RESPONSE

A Public Domain Approach to Free and Open Source Software?

GREG R. VETTER*


TABLE OF CONTENTS

I. INTRODUCTION .................................................................................................................. 8
II. THREE ACCOUNTS OF COST AND CODE................................................................. 11
III. THREE CONSIDERATIONS FOR OPEN INNOVATION IN SOFTWARE....... 15
IV. CONCLUSION..................................................................................................................... 17

I. INTRODUCTION

While other scholars have described the need to reduce non-beneficial anticommons effects in the scattershot licensing approach currently supporting free and open source software (FOSS),1 Professor Clark Asay’s article, A Case

---

*Professor of Law, University of Houston Law Center; Law Foundation Professor, Co-Director, Institute for Intellectual Property and Information Law; biography available at: www.law.uh.edu/faculty/gvett. My background includes a Master’s degree in Computer Science and nine years full-time work experience in the software industry.

1 See, e.g., Robert W. Gomulkiewicz, Open Source License Proliferation: Helpful Diversity or Hopeless Confusion?, 30 WASH. U. J.L. & POL’Y 261, 279–82 (discussing license proliferation). In this article, Professor Gomulkiewicz discusses his creation of the Simple Public License (SimPL) in order to illustrate the possibility of a less scattershot world of FOSS licensing. Id. at 264–70; see also Ctr. for Advanced Study & Research on Intellectual Prop., Simple Public License (SimPL), U. WASH. SCH. LAW, http://www.law.washington.edu/casripl/license (last updated Apr. 27, 2012). Arti Rai also posits a theoretical basis from which one might understand the problems with FOSS license proliferation—an anticommons-type effect. See Arti K. Rai, “Open Source” and Private Ordering: A Commentary on Dusollier, 82 CHI.-KENT L. REV. 1439, 1440–41 (2007).

The FOSS ecology includes various groups, and those affiliated with the open source (as opposed to free software) approach are particularly concerned about license proliferation. Greg R. Vetter, Exit and Voice in Free and Open Source Software Licensing: Moderating the Rein over Software Users, 85 OR. L. REV. 183, 187–88, 196–97, 264–65 (2006) [hereinafter Vetter, Exit and Voice].
for the Public Domain,

while somewhat focused on open-innovation communities more generally, comprehensively details these effects and their costs. His critique of the transaction costs and other issues relating to vivacity within development groups and corporate use of FOSS provides a unique contribution to the literature. His call to shift the legal regime underlying FOSS to a public domain basis, jettisoning an IP rights based approach, with support from hypothesized new law to enable a strong public domain dedication, resonates with the rich literature studying the public domain and commons-based approaches to innovation.

The framework proposed by Asay is intended to be generally applicable, but is primarily rooted in FOSS considerations as compared to content communities such as Creative Commons. In this Response, I will focus on the policy aspects of Asay’s analysis and proposal as they relate to software as an ecology and a technology.

This Response proceeds in two parts. First, with Asay’s cost description in relief, Part II discusses costs, code, and value. Second, in light of Asay’s proposed shifting of FOSS to a public domain approach, Part III discusses: the political economy of regime shifting and alternative regimes, institutional design risk for the public domain, and influences for openness in software.

At bottom, I have myself, in an early article, called for a new statutory basis to support the FOSS movement. The decade since has provided some progress in FOSS licensing, but mostly from the perspective Asay notes: transactional licensing lawyers are increasingly knowledgeable about how to use FOSS. Inbound and outbound licensing of FOSS for commercial concerns is much more common. While the development of FOSS licensing law in the courts remains slight over the last decade, that doesn’t necessarily mean that the current FOSS “system”—if one can call it that—isn’t working.

---

3 Id. at 768–75.
4 The literature on the public domain is far too vast to catalog in this Response, in which I am purposefully directed to keep footnoting to a minimum. Professor Asay’s article provides many of the classic sources. One source worth noting, however, because it arrives at recommendations similar to Asay’s insights but based on the study of rhetorical narrative within social change, is an article by Professor Jessica Silbey: “[D]iscreding or revolutionizing the language of traditional intellectual-property law is necessary to facilitate a sea change in our intellectual-property relations.” Jessica Silbey, Comparative Tales of Origins and Access: Intellectual Property and the Rhetoric of Social Change, 61 CASE W. RES. L. REV. 195, 199 (2010).
6 The FOSS licensing movement uses several copyright-based licensing techniques to emphasize source code transparency and, for many licenses, requires subsequent development to occur under the same or a similar FOSS license. Sometimes the licenses include anti-royalty provisions for ongoing software use. At other times, they require extension of the FOSS terms to closely intermixed software, an effect sometimes called the
The lack of disputes in courts might mean most organizations most of the time are certain enough, even if not perfectly certain, about clearing rights for inbound FOSS code. As compared to inbound licensing of proprietary software, a different bundle of advantages and disadvantages encircle FOSS use for commercial concerns. Whether these cost considerations truly prompt a need, as Asay argues, to nudge FOSS into a public domain regime, remains to be seen. However, on the other hand, one can envision a public domain act, inspired by Asay’s article, but with additional details to fill in the gaps he intentionally left, where FOSS licensing would be more systematic and certain.

Asay’s broad sketch of a law enabling public domain dedication includes: a public domain symbol one could attach to works; the recognition that the provenance of code might still be an issue, and thus that perhaps there should be penalties for those who might strip the symbol from the works;\(^7\) and positing that the core effect should be:

A public domain approach, therefore, would need to effectively override any automatic copyright rights, waive any patent rights (both with respect to any patent rights already obtained as well as prospective rights), and relinquish any remedies that come with either. Trade secret rights, if any, would be inapplicable as soon as the rights holder released the software or content to the public. Arguably waiving any trademark rights is not only unnecessary but inadvisable . . .\(^8\)

The patent law characteristics of the sketched act would be to clear rights in patents held by those users or developers involved with the software, which still leaves the risk of patents held by others.\(^9\) With any regime of rights or immunities, the details matter. With the uncertainties encircling FOSS licenses reciprocity requirement. The primary basis of a FOSS license is typically copyright law, although some FOSS licenses include provisions relating to patent law. Often FOSS licenses are classified into types. One type, attribution-only licenses (sometimes called BSD-style licenses), generally allows any use of the software, even in proprietary products without source code, so long as attribution is given. Another type, the “copyleft” or reciprocal license, typically has several requirements: (1) royalty-free software use; (2) available with source code; (3) distributable in modified or unmodified form; (4) with recipient users and redistributors granting a copyright license to other recipients for any added development; and (5) with all these conditions applying to future generations of the software upon redistribution with or without modification, including modifications that intermix other software. In this Response, unless a specific license or license type is named, I do not intend to single out any specific license.

\(^7\) Asay, supra note 2, at 784, 794–95, 801–02.

\(^8\) Id. at 784.

\(^9\) Id. at 794, 801–05. Asay provides a variety of structural arguments why, under a public domain act approach, there might be less risk of external threats from patents held by others outside the open innovation community involved with the software. My purpose is not to delve into those detailed considerations, but to note that the proposed public domain act does not seek to affirmatively clear rights in patents held by such others.
in the areas of copyright and contract and patent law, a regime of public domain dedication rights and immunities could add certainty to the law. The patent risk from those outside the open innovation community, however, remains, and as Asay notes, is an artifact of the structure of U.S. patent law. More specifically, there is no independent developer defense in U.S. patent law, where such a defense would fundamentally shift the patent regime to one where copying from the patent inventor or holder was a part of proving infringement. Thus, as things stand currently, or under Asay’s proposal, the patent system remains a threat to open innovation in FOSS.

With this review of Asay’s public domain proposal in view, the next part discusses his taxonomy of costs for code under the current FOSS licensing approach versus his proposed act. I add a third scenario: proprietary software.

II. THREE ACCOUNTS OF COST AND CODE

Typically, organizations obtain software either to use it to support internal operations, or put it into a product that the organization distributes externally. Occasionally, both uses apply. The concept of external distribution should be taken broadly. It can include selling a software product where a component of the software product is third-party software with an inbound license allowing the external distribution. Or it can include selling a device with software inside it. Or it can include software delivered as a service, an approach that has become synonymous with the term “cloud computing” or merely “the cloud.”

While acknowledging that FOSS needed to implement an intellectual property-based licensing regime to emerge and grow to its current critical mass, Asay posits an alternative world where the next phase of FOSS would be best

---

10 There are various issues of doctrine not well settled with FOSS licensing. See David McGowan, Legal Implications of Open-Source Software, 2001 U. ILL. L. REV. 241, 268, 289–302 (discussing doctrines related to a variety of issues, including assent, privity, term, termination, and assignment).
11 Various formulations of an independent developer defense are possible. See Oskar Liivak, Rethinking the Concept of Exclusion in Patent Law, 98 GEO. L.J. 1643, 1648–49 (2010) (comparing traditional patent law exclusivity in the United States to exclusivity that does not reach independent developers of the same technology); Roger Milgrim, An Independent Invention Defense to Patent Infringement: The Academy Talking to Itself: Should Anyone Listen?, 90 J. PAT. & TRADEMARK OFF. SOC’Y 295, 297 (2008) (discussing various practical issues concerning two proposals from the legal academy to consider independent development as a defense in patent law); Samson Vermont, Independent Invention as a Defense to Patent Infringement, 105 MICH. L. REV. 475, 479 (2006) (arguing that patent protection need not necessarily cover independent inventors in order to provide a sufficient incentive to invent).
based on public domain dedications. A major part of the argument for this regime change is that costs relating to using the software will be lesser in the dedication regime versus the current scattershot FOSS licensing “system.”

The taxonomy of costs include: inbound licensing costs, internal management costs, costs related to mergers and acquisitions of companies with licensed rights in software, and outbound licensing costs. I will call these, collectively, deployment costs. Asay’s rich and detailed account of deployment costs for FOSS is impressive, comprehensive, and a unique contribution to the literature. The focus is FOSS deployment in a corporate or commercial setting. Non-profits, or even individual developers, who implement inbound and outbound licensing of FOSS will often also need to account for all of these costs, with perhaps the exception of merger and acquisition costs.

Based on the detailed review of deployment costs, Asay’s central proposition is that FOSS license proliferation has anticommons effects, making it more costly to clear rights for use of FOSS. In addition, Asay posits that two central mechanisms of some FOSS licenses, reciprocity and attribution, could be deemphasized. Attribution might be better left to non-licensing technological mechanisms,14 while reciprocity is perhaps no longer a needed license feature.15 Both of these add to the deployment costs for FOSS.

Discussing costs leads one to consider value. The FOSS movement has been valuable, and has generated social value above and beyond the value created by it within information technology. One might organize the discussion of value temporally, considering three modes of software innovation: before we had FOSS, we had proprietary licensed software, which has costs and gives value. Then, we added FOSS licensing, and its value/cost proposition has evolved over time. Next, Asay posits, we might replace FOSS with public domain dedications for open innovations. Practically speaking, if a public domain dedication act were implemented, it might allow for a second alternative mode of open innovation in software rather than replace FOSS licensing, although this would depend on the details of the act. For the rest of this Part, this Response considers all three modes to draw some higher-level observations.

The public domain dedication mode of open innovation in software, Asay posits, would have lesser costs than currently prevalent FOSS licensing. This is clearly plausible, although whether that is the result depends on, among other factors, the details of the dedication act. Asay’s account is realistic: there are still costs in a dedication regime. For example, inbound licensing still must care about the provenance of the code: someone might take the source code for proprietary licensed software, put the dedication symbol on it, remove all

14 Specifically, Asay notes that a recently popular web site for posting code, called GitHub, might expose attributions more beneficially than burying the attributions in license notice files. Asay, supra note 2, at 777–78.
15 Id. at 778–82.
identifying comments and other information, and put it into a public repository.\textsuperscript{16}

While not a part of Asay’s treatment, one wonders whether deployment costs for FOSS licensing will improve in the future. The costs could improve because lawyers and licensing professionals could increasingly learn how to deal with FOSS licenses (they have already learned much about dealing with them, which is why Asay’s account of the costs to do so is richly detailed).\textsuperscript{17} Perhaps some real headway will occur against license proliferation, leading to a FOSS licensing system that is more systemized.\textsuperscript{18} But improving the FOSS system is not necessarily Asay’s project. His thought experiment is to highlight the system against a counterfactual alternative.

The oldest mode of software commercialization is proprietary licensing.\textsuperscript{19} Its deployment costs are different than FOSS, but many of the same considerations apply.\textsuperscript{20} To extend the cost perspective, there are costs involved with inbound licensing of proprietary software that are much higher than the

\textsuperscript{16} Id. at 765–68.

\textsuperscript{17} The proliferation of FOSS licenses leads Asay and others to posit an anticommons effect. See supra note 1. Others, however, when considering possible counter-effects for an anticommons, note that when the law and its practitioners learn how to arrange affairs, “transaction cost optimism” can be a reason to doubt whether the anticommons effect is as negative as feared. See Robert P. Merges, \textit{Software and Patent Scope: A Report from the Middle Innings}, 85 Tex. L. Rev. 1627, 1630–31 (2007).

\textsuperscript{18} The inspiration of the Creative Commons set of licenses is an obvious example, but these are for mere content, not entangled code. Asay discusses the Creative Commons example given that his treatment of the public domain act is envisioned to include more than software. See supra note 1 and accompanying text.

\textsuperscript{19} When I use the terms “proprietary software” or “proprietary licensing” I mean either a software product or other licensed software in which the licensor requires royalties for use of the software, either in an up-front payment or on an ongoing basis. In addition, the term means software with narrow permissions for use and for which the source code is not disclosed or made available to the end users in the typical license transaction. In essence, I use the term to refer to the historically dominant practices used by software providers to protect and license their computer programs. While source code is rarely distributed to end user licensees of proprietary software, oftentimes one proprietary software provider will license source code to another such provider for purposes of the licensee provider incorporating the licensor’s technology into its software product.

\textsuperscript{20} Heather J. Meeker, \textit{The Open Source Alternative: Understanding Risks and Leveraging Opportunities} 56–57 (2008). Meeker elaborates on the due diligence process for inbound licensing as follows:

The more inbound licenses apply to your code base, the more complicated the due diligence task. This process, and the complications engendered by it, is not specific to open source. This process receives so much attention in the open source world because the rise of open source has caused developers to have better, and less expensive, access to code—and to combine code from many different sources. . . . The only thing that has changed due to open source is the volume of diligence to be conducted, not the basic nature of it.

\textit{Id.}
other two modes, FOSS or public domain. First, there will be a royalty to pay. Second, finding the software and evaluating its functionality and quality is oftentimes more difficult. Third, if the inbound license for a proprietary component will not include source code rights, often the licensee will want to arrange source code escrow, a transaction cost not needed for the FOSS or public domain modes.

Extending the cost discussion to the proprietary mode is to make the point that the value/cost proposition for each mode is unique. Moreover, from a policy perspective, value should account for more than the value to the software seller and the value to the software user, but also the social value for the product in use in society. For example, after FOSS the availability of source code for study and learning was much greater than in the previous regime where software was dominated by proprietary licensing.

If cost considerations prompt a nudge of FOSS from a licensing system to a public domain system, those cost considerations should operate in a larger assessment of value. Here, empirically, there is both no way to have a solid answer, and a need for more study. To illustrate the role social value should play in the analysis, however, I will stylize a hyper-simplified model against the question.

For the model, assume that all types of software have the same value influences, where value has three components: value to the licensor, value to the user/licensee, and social value. Costs are only assigned to the licensor as deployment costs. Technically, there should perhaps be a breakout of user licensee costs, but the primary purpose is to demonstrate the influence of adding social value to the policy analysis. User licensee costs are likely proportionally smaller than the licensor’s overall costs to deploy the software, and thus are blended into the model only indirectly through the user/licensee value amount.

---

21 The search costs are significant because the source code isn’t publicly available. Software technologists searching for a software component to inbound license into a company’s software product might know of several potential component suppliers. But the evaluation often is not as easy as it is with FOSS, where one can, in a matter of minutes, download the code and begin evaluating its quality, efficiency, and fit for the product (the ability to examine the source code is an important part of the quality evaluation). The transparency of publicly available source code in FOSS provides an interoperability benefit as well as an ease of evaluation benefit.

22 Vetter, Exit and Voice, supra note 1, at 228–30.


24 Technically, there should perhaps be a breakout of user licensee costs, but the primary purpose is to demonstrate the influence of adding social value to the policy analysis. User licensee costs are likely proportionally smaller than the licensor’s overall costs to deploy the software, and thus are blended into the model only indirectly through the user/licensee value amount.
### Table 1: Value Factors

<table>
<thead>
<tr>
<th>Value Factor</th>
<th>Proprietary</th>
<th>FOSS Licensing</th>
<th>FOSS Public Domain Dedication Act</th>
</tr>
</thead>
<tbody>
<tr>
<td>Licensor deployment cost</td>
<td>(2)</td>
<td>(3)</td>
<td>(1)</td>
</tr>
<tr>
<td>Licensor value</td>
<td>7</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>User/Licensee value</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Social value</td>
<td>11</td>
<td>14</td>
<td>12</td>
</tr>
<tr>
<td>Total Value</td>
<td>18</td>
<td>18</td>
<td>18</td>
</tr>
</tbody>
</table>

In the model’s account, I implement Asay’s deployment cost reduction hypothesis for public domain FOSS (a unit of 3 versus a unit of 1). Licensor value is likely the same for both types of FOSS, but value to the proprietary software vendor is likely greater than either FOSS mode. I establish the user/licensee value as the same (a unit of 2) across the board.25

The social value contribution is the key insight for the table: Asay’s account assumes that social value for FOSS will remain the same or increase under a public domain dedication approach. My purpose in setting it lower is to illustrate that if that does not happen, then the policy nudge to a new mode of FOSS is not necessarily warranted. For example, in the table, if the unit of 12 for social value for the public dedication approach was 13, then this suggests a shift (the total value for the dedication approach goes to 19), assuming the regime transition costs are not too great.

This stylized, hyper-simplified example serves only to highlight the important questions to be studied. The table above is likely different in value and cost structure for different types of software in different markets. A complexity to the assessment quickly emerges. The next Part will discuss some aspects of that complexity, both as to the regime transition issue and the comparative social value.

### III. Three Considerations for Open Innovation in Software

There is a fundamental question whether the proposed public domain act is regime shifting or regime competition. If an alternative, as a non-IP-based mode to approach open innovation in FOSS, it would be able to compete based on

---

25 I set the user/licensee value as the same across the board for several reasons. First, the user might pay less, or perhaps pay nothing, for the FOSS modes, thus they should be the same. On the other hand, FOSS still mostly occupies technological niches. Thus, even though FOSS is often “free” as to licensing royalty costs, there are sometimes other costs, such as perhaps needing technologists to provide services to implement the FOSS code. These considerations vary by type of software and from FOSS project to FOSS project. In sum, it is not implausible that the costs for FOSS at the user level sometimes approach the royalty laden proprietary software costs.
mindshare and perceived value. The former option, regime replacement or shifting, is likely practically impossible, but seems to be the focus of Asay’s thought experiment. Outright regime replacement would conflict with the existing copyright in tremendous amounts of code. Whether replacement or shifting or a new alternative, federal law will need to be enacted. This raises the first of three points for this Part: the political economy issues of the proposed public domain act.

The political economy issues for lawmaking in general are familiar to many readers. These issues have arisen in intellectual property lawmaking, both domestically and internationally.26 The content and software industries that depend on copyright will attentively monitor and seek to influence the public domain act so as to minimize any hypothesized negative impact its existence might have on revenues or market influence. This may result in an act that still fulfills its purpose, or it may result in an act where the social value of public domain dedications cannot reach the levels currently enjoyed with FOSS licensing. The scenarios one might discuss under this topic are numerous, but I will leave the point without a specific example. Instead, I make the classic point from public choice theory about organized and focused interest groups and their success in legislative lobbying as compared to diffuse interests. This has always been a challenge for the public domain—the public’s interests are often underrepresented in Congress.

A second issue for open innovation in software is also a risk for the efficacy of the public domain act: the end-user license agreement (EULA). Enforceable in contract when a “click” is obtained as an indication of assent, it has been used to limit various practices that enable software to have increased interoperability.27 As federal law, the public domain act would need strong contract preemption provisions to help ensure that the EULA does not become the barbed wire re-enclosing the dedicated materials.

The final point for this Part is also potentially expansive: what forces beyond the deployment cost model influence openness in code. Consider, for example, the reciprocity requirement and the concept of critical mass. Some FOSS licenses require reciprocity,28 raising the question about the impact of contributions to that project. On the one hand, many programmers, due to a strongly felt value system, will only contribute to projects with licenses requiring reciprocity. On the other hand, reciprocity limits the code’s ability to


27 DOUGLAS E. PHILLIPS, THE SOFTWARE LICENSE UNVEILED: HOW LEGISLATION BY LICENSE CONTROLS SOFTWARE ACCESS 4–6, 26–35, 50–51 (2009) (remarking, concerning the typical terms of EULAs: “The right to engage in fair use is overridden. The right to circumvent technological measures for interoperability is overridden.”).

28 In Creative Commons licensing, the label used is “Share Alike” for the reciprocity feature. See About the Licenses, CREATIVE COMMONS, http://creativecommons.org/licenses (last visited Mar. 1, 2014).
be blended into certain other projects, particularly if those projects have components that are proprietary software. But reciprocity is also a reflection of core aspects of FOSS as a social movement. Without its motive force as a social movement, FOSS might not have ever become an alternative pathway for developing and deploying software. The anticommons effect of many licenses has a flip side—a diversity of licenses reflects a diversity of values and diversity of groups deciding to contribute to FOSS projects. This point holds sway even under the recognition that increasingly FOSS contributions are subsidized or part of commercially oriented FOSS support. Those enterprises that support FOSS have employees who believe in FOSS values and the enterprises are by no means monolithic in their perspective.

IV. Conclusion

Richard Stallman started the free software movement by cleverly realizing he could use property rights, that is copyright, to control what he saw as an immoral practice: keeping source code secret. That insight led to the realization that free distribution with available source code was also achievable with copyright-based licensing. The success of Stallman’s General Public License (GPL) version 2.0 was the result. Later, the open source camp blended into the movement that I typically refer to as FOSS. At bottom, the FOSS movement, and the licenses that underlie it, represent a private, collective, although highly disjointed and distributed, effort to standardize property forms. A law that would standardize the public domain for software can be viewed as a government-supplied form of anti-property, just as FOSS licenses, particularly those of the GPL-style, are a type of anti-property compared to proprietary software.

Whether private standard setting (a bunch of groups over time making a bunch of FOSS licenses) or government-supplied property forms (Asay’s public domain act) are preferable is a question that has been posed more generally in property theory. One of the key articles to treat this question theoretically inclines toward an answer that supports Asay’s insight. The article uses the term “numerus clausus” to indicate the principle whereby the forms of enforceable property interests are limited.29

[T]here are theoretical and empirical reasons to doubt that private ordering will do better than legal constraints in standardizing the basic building blocks of property rights. . . .

The question would be whether private provision of standardization would lead to enough benefits in terms of flexibility to be worth the cost of the

---

29 Thomas W. Merrill & Henry E. Smith, Optimal Standardization in the Law of Property: The Numerus Clausus Principle, 110 YALE L.J. 1, 4 (2000) (“In the common law, the principle that property rights must conform to certain standardized forms has no name. In the civil law, which recognizes the doctrine explicitly, it is called the numerus clausus—the number is closed.”).
remaining confusion and the legal intervention needed to support it. If, as we argue, the frustration costs from a fairly severe degree of standardization through the *numerus clausus* are not that large, there is little reason to think that private standards will be more cost-effective than those supplied by the government. Moreover, if one set of basic building blocks will do for most purposes, government provision of such a system is likely to be characterized by significant economies of scale and scope.\textsuperscript{30}

Asay’s interesting and provocative suggestion to supplant FOSS licenses with a *public* domain act starts from a more practical perspective than the theoretical vanguard of the *numerus clausus* principle. But the suggestion’s insight resonates with that principle. On the other hand, however, and despite Asay’s instructive account of FOSS deployment costs, there are reasons to be concerned about the political economy both of a move from FOSS to the public domain approach, and whether the non-IP regime of that approach disarms those who built the FOSS movement. Even as a prodigious software developer with many feats attributed to him, Richard Stallman’s greatest “hack” may be turning IP rights against the forces of secrecy for source code. This is the unique potency of property rights. The likely outcome of a public domain dedication act is not to supplant FOSS licensing, but to offer an alternative regime. That regime’s success will depend on its costs and social value over time as it competes with the two other modes of software innovation.

\textsuperscript{30} *Id.* at 50–51 (footnote omitted).