Boston College Law Review

Volume 48
Issue 1 *Owning Standards Symposium*

Article 9

1-1-2007

Open Source Licensing and Scattering Opportunism in Software Standards

Greg R. Vetter

OPEN SOURCE LICENSING AND SCATTERING OPPORTUNISM IN SOFTWARE STANDARDS

Greg R. Vetter*

Abstract: Despite their beneficial influence on interoperability and markets, problems of detrimental opportunism occur with technology standards, including standards implemented in software, which this Article calls "Software Standards." Inspired by new perspectives on the study of semicommons in the history of real property, this Article contemplates the substitutability of free and open source software ("FOSS") for traditional standard-setting approaches. Standards are analogous to semicommons, where public and private use interact, raising the possibility of opportunistic influence on the Software Standard to increase private gain at the expense of the public benefit in a more uniform standard. With its source code disclosure requirement, FOSS shifts and dampens this opportunism, although various limits influence the reach of its effect. The political economy around a standard will express itself differently under a FOSS implementation, and clearing intellectual property rights in the standard is no more certain than under the traditional standard-setting approach.

Introduction

Technology standards are indispensable to the modern world, but they come with the risk of strategic and opportunistic behavior by those involved in creating the standard and implementing products that conform to it. Standards provide both public and private benefit. This division enables counterproductive opportunism, which involves private gain at the expense of a more ubiquitous or uniformly imple-

* Assistant Professor of Law, University of Houston Law Center (UHLC); Co-Director, Institute for Intellectual Property and Information Law; biography available at www.law.uh.edu/faculty/gvetter. My background includes a master's degree in computer science and nine years of full-time work experience in the software industry. My thanks to UHLC students Stacey Reese, Jeffrey Tom, and Nivine Zakhari for research assistance. For helpful comments and discussion, I thank Darren Bush, Craig Joyce, and participants at the Owning Standards symposium held in March 2006 and sponsored by the *Boston College Law Review*, particularly Michael Carroll, who moderated the open source panel at the symposium.

mented standard. Thus, opportunistic behavior seeks private gain at the expense of the public benefit in the standard. Such opportunism can occur during creation of the standard, and also post-creation, as suppliers implement the standard. It can involve skewing the standard to a firm's particular capabilities or can involve proprietary intellectual property rights in the standard. Dampening such opportunism is a policy concern for standard creation, revision, and implementation.

A typical response to dampen opportunism is norm enforcement, where a community or authority enforces practices that inhibit counterproductive opportunism. For standards implemented in software, which this Article calls "Software Standards," lessons from the semicommons in the history of real property suggest an alternative approach—technological "boundary" rearrangement.[1] Software Standards are a type of technology semicommons. A beneficial technological boundary rearrangement may inherently occur in this semicommons if Software Standards are implemented as free and open source software ("FOSS").[2] This potential occurrence suggests the potential substitutability of FOSS for the approach of traditional standard-develop ment organizations ("SDOs").

The historical semicommons referenced above is the open fields system in England.[3] Under this system, each landowner held scattered, irregular-shaped strips of land. Aggregately, the strips were

[1] *See infra* notes 43–47 and accompanying text.

[2] This Article accepts as a premise, without claiming that it is fully demonstrated, that the FOSS licensing system works well enough to support the five primary conditions this Article uses to give meaning to the FOSS label: (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 patent license to other recipients for patents controlled by the recipient that cover the software; 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. There are various issues of doctrine that are not well-settled with FOSS licensing. *See* David McGowan, *Legal Implications of Open-Source Software*, 2001 U. Ill. L. Rev. 241, 289–302 (discussing doctrinal issues related to a variety of matters, including assent, privity, term, termination, and assignment). Some FOSS licenses include additional patent provisions, extending the treatment beyond condition four in the list above by eliminating or suspending a recipient's right to use the software if the recipient asserts patents against the software. These are sometimes called "patent peace" clauses.

My definition of FOSS licensing excludes attribution-only licenses. Thus, I focus on what some call "copyleft" FOSS. Attribution-only licenses are sometimes called BSD-style licenses. They generally allow any use of the software, even in proprietary products without source code, so long as attribution is given. Finally, unless a license is named, this Article does not intend to single out any specific license. FOSS licensing is taken as a system defined by the five conditions above.

[3] *See infra* notes 43–47 and accompanying text.

productive on a public scale as a commons for grazing, but individually their private use was agricultural. This boundary manipulation of ownership into strips was theorized to decrease the opportunistic possibility for any individual landowner to gain to the detriment of the others' use or the public use of the land.[4] During certain periods, the historical semicommons favored boundary rearrangement over norm enforcement to inhibit opportunism.

As technological semicommons, SDOs often adopt norms to govern their activities. Positive law, such as antitrust, also applies to SDOs. Norm enforcement may cover procedures for deliberation and decision making to allow SDO participants to check each other, or it could cover practices to clear intellectual property rights involved with the standard. The SDO approach to inhibiting opportunism is synonymous with norm enforcement.

When promulgating a Software Standard, SDOs typically issue public documents describing the standard. Companies often implement the standard in private code, however, using a proprietary approach to software licensing that keeps source code secret and distributes only object code. The SDO approach facilitates secret implementation, which in turn facilitates opportunism.[5]

Like SDOs, FOSS development needs to inhibit counterproductive opportunism and clear intellectual property rights for use of the software. SDOs and FOSS have overlapping approaches to inhibiting opportunism, with similarities and differences. FOSS licenses require a fully public implementation because the software must disclose source code, and not just provide object code, among other important conditions.

This Article proposes the potential substitutability of a FOSS peer production process[6] for the SDO approach to Software Standards. This change is akin to replacing norm enforcement with scattered strips of property in the historical semicommons example. The FOSS approach provides technological boundary rearrangement under its license conditions that require freely redistributable source code for successive generations of the software. In other words, FOSS could

[4] *See* Henry E. Smith, *Semicommon Property Rights and Scattering in the Open Fields*, 29 J. LEGAL STUD. 131, 162–65 (2000).

[5] *See infra* notes 13–22 and accompanying text.

[6] *See* Yochai Benkler, *Coase's Penguin, or, Linux and the Nature of the Firm*, 112 YALE L.J. 369, 372–78 (2002) (discussing a generalized production theory of aggregating creative inputs consistent with and inspired by FOSS).

function as a "direct to code" standard. This is similar to what the result would be if a FOSS product became a de facto standard.

A complex inquiry then emerges: what conditions would be conducive for an SDO to promulgate a Software Standard directly as FOSS code? To analyze this question, Part I discusses intangible ownership in standards, as well as creation-process and implementation opportunism with Software Standards.[7]

Part II examines norm enforcement for SDOs and FOSS.[8] Although a variety of norm examples exists for SDOs, this Article uses the Standards Development Organization Advancement Act of 2004 (the "SDOAA" or "Act") to focus the inquiry.[9] The SDOAA limits antitrust liability in certain ways for an SDO "while engaged in a standards development activity."[10] To qualify for this protection, the SDO must run a consensus process.[11] Thus, the enforcement pressure is through an antitrust law benefit. FOSS norms are enforced through the license and through a potent ideology coupled with software community practices.

Part III presents the SDO and FOSS approaches to Software Standards.[12] FOSS's royalty-free source code disclosure requirement brings a different mix of benefits and costs to the initial and downstream developers and users. It induces an opportunity cost for participants in foregone potential for some private-code economic rents. Comparing this situation to the semicommons of a standard, one sees that the new

---

[7] *See infra* notes 13–24 and accompanying text.

[8] *See infra* notes 25–42 and accompanying text.

[9] Standards Development Organization Advancement Act of 2004 §§ 101–108, 15 U.S.C. §§ 4301–4305 (Supp. IV 2004).

[10] *Id.* § 4302(2). The SDOAA (1) de-trebles damages for SDOs that file an identifying notification with the federal government disclosing the nature and scope of activities to develop or promulgate a voluntary consensus standard, (2) requires a rule of reason analysis for SDOs, and (3) shifts attorney fees. *Id.* §§ 4302–4305.

[11] Following an Office of Management and Budget Circular referenced by the SDOAA, this Article uses the term "consensus standard" for the output of voluntary, open organizations with inclusive processes. OFFICE OF MGMT. & BUDGET, EXECUTIVE OFFICE OF THE PRESIDENT, CIRCULAR NO. A-119, FEDERAL PARTICIPATION IN THE DEVELOPMENT AND USE OF VOLUNTARY CONSENSUS STANDARDS AND IN CONFORMITY ASSESSMENT ACTIVITIES § 4 (1998) [hereinafter OMB CIRCULAR], *available at* http://www.whitehouse.gov/omb/circulars/a119/a119.html.

Particularly in information technology, there is an increasing use of consortia as an alternative to the consensus approach. Consortia differ from consensus SDOs in a variety of ways, including a willingness to use expedited process rather than the intricate rules of the consensus organizations. Carl F. Cargill, *The Sisyphus Agenda: Standardization as a Guardian of Innovation*, *in* THE STANDARDS EDGE: DYNAMIC TENSION 31, 34–35 (Sherrie Bolin ed., 2004).

[12] *See infra* notes 43–55 and accompanying text.

FOSS boundary diminishes opportunities for private gain at the expense of the common good under the theory that FOSS facilitates greater adoption of a more uniform and uniformly implemented standard. The toll for crossing the boundary into the standard when intangible rights are involved is reduced from the SDO's oft-used reasonable and nondiscriminatory ("RAND") royalty to the FOSS decentralized royalty-free approach, with a corresponding reduction in affiliated transaction costs. Shareable FOSS code might reduce duplicated efforts to implement the standard. On the other hand, the FOSS approach might affect standard adoption rates if participants wait for someone else to implement the shareable code. Moreover, the FOSS license might need specific provisions for use in the standards context so that participants do not fear a requirement to disclose product source code or over-commit patent rights. If nothing assuages these concerns, the FOSS approach might attract a smaller participant group, potentially bringing a corresponding reduction in the intangible rights cleared for the standard.

Also important are comparative fragmentation possibilities. Private parties sometimes try strategically to extend or customize a standard, perhaps increasing its technical features, but at the expense of uniformity. The FOSS fragmentation problem is called "forking," which has the potential to cause related problems, but differs in that distributed forked FOSS code can be used by anyone to gain whatever advantages may exist in the forked implementation. Thus, fragmentation of FOSS code may not inevitably lead to a permanent schism in the standard and may allow an easier escape from a suboptimal equilibrium with the technology.

In conclusion, this Article emphasizes the implications for future standards development. Often, a standard can be partitioned from another layer of technology and implemented as working software. When this is so, the FOSS approach should be considered as an alternative to the traditional standards-development process.

I. Strategic Behavior with Software Standards

Standards present problems of moral hazard. The incentives to game the creation process to lock in competitive advantage are in tension with the goal of creating a viable, widely adopted standard. Even after origination, a variety of strategic behaviors regarding the standard may advance one implementer's competitive position at the expense of the common benefits underlying standardization. Some of these opportunistic tendencies are exacerbated with Software Standards. After reviewing the question of intellectual property ownership in a standard, this Part discusses the opportunism scenarios to illustrate the need for countervailing forces to inhibit them.

A. *Intangible Ownership and the Dichotomy of Standards*

Ownership rules for a standard and its implementations depend on the standard's creation process. A de facto standard[13] may bring private ownership to the entity or entities that develop the product or platform. A de jure standard developed under a consensus process may reduce control from intellectual property-based ownership rules. Standards involving either category may induce firms to pool rights to intangibles, such as patents, in order to reduce infringement risk through cross-licensing. For all of these examples, this Article uses the term "ownership rules" broadly to include licensing conditions for the standard or its implementations.

Unless something governs the input rights and the rights created or contributed during the standard creation process, these rights leave open the possibility for property-like control over the standard.[14] The typical rights at issue arise from trade secret, trademark, copyright, and patent law. Rights owners may have the opportunity to exclude use of the standard through exclusionary rights and control its disposition. The strongest instance of this is often a de facto standard, one example being Microsoft's Windows operating system for desktop computing. But even a de jure standard developed through a participatory consensus process may retain property-like features, such as, for example, requiring payment for use under RAND terms.[15]

[13] Delineating precisely when standards are de facto versus de jure is not central to my analysis. An additional point for software-implemented standards is that they are often layered: a de facto standard may depend on a de jure standard which may incorporate elements of other de facto standards, and so on. As a definitional matter, de facto standards are those that are not de jure, but de jure standards are harder to cabin. They may flow directly from governmental authority, quasi-governmental entities, or organizations whose output is approved or incorporated by the government in some way. *See* Mark A. Lemley, *Intellectual Property Rights and Standard-Setting Organizations*, 90 Cal. L. Rev. 1889, 1898 (2002); Janice M. Mueller, *Patent Misuse Through the Capture of Industry Standards*, 17 Berkeley Tech. L.J. 623, 633–34 (2002).

[14] If there are patents existing prior to the standard but not discovered during the creation process, there is also the possibility of post-creation control over the standard by third parties not involved in its creation. If non-participant patents surface during standard creation, the participants could seek a license or direct the standard away from the patented technology.

[15] Joseph Scott Miller, IP Policy as Governance Structure: A Fresh Look at the RAND Promise 4–6 (Jan. 10, 2006) (unpublished manuscript, on file with author).

RAND licensing is a liability-rules approach often used for de jure or consensus standards. It illustrates standards' function to provide both public and private benefits. Under an optimistic view, the SDO participants anticipate that collaboration will pay off in a novel technology platform generating new opportunities for the participants and third parties, despite the potential for induced limitations on appropriability for any intangible assets contributed to the standard.[16] Under a pessimistic view, intellectual property rights may underpin opportunism by SDO participants during standard creation and thereafter, even if the RAND licensing promise is made.

Ideally, participants clear their own intellectual property rights against the standard. This does not eliminate, however, the possibility that third parties, in particular those holding patents, will not have leverage over the standard through their own intellectual property rights. Independent development after the patent issues is not a defense to infringement. Thus, third party patent-holders may threaten both de facto and de jure standards.

FOSS licensing is a hybrid of the various ideas discussed in this Section. It spans the classic division of de facto versus de jure standards. FOSS, such as the Linux kernel or its license, the General Public License (the "GPL"), may achieve de facto standard status. FOSS developers often interact over code and its licensing, in ways similar to participants of SDOs, hoping to spawn de jure, consensus, or consortia-based standards. Consortia SDOs, with less intricate and open procedures than consensus processes, would fall somewhere between consensus SDOs and FOSS if one arrayed these on a continuum.[17] FOSS licensing began as a copyright cross-licensing scheme with conditions applying upon a distribution, but has expanded to include patent licenses granted to recipient-users of the software by those who distribute it. Like a typical SDO, a FOSS group wants a license that clears intellectual property rights to the greatest extent possible to facilitate maximum spread of the software.

[16] *Id.* Historically, the SDO arrangement created opportunities for strategic behavior. *See* ABA SECTION OF ANTITRUST LAW, HANDBOOK ON THE ANTITRUST ASPECTS OF STANDARDS SETTING 14–19, 56–67 (2004) [hereinafter ANTITRUST HANDBOOK]. Related to this history is the fact that scholars have critiqued the methods by which SDOs disgorge and clear intangible rights for the inputs and license the standard for use post-creation. *See* Lemley, *supra* note 13, at 1904–06 (reviewing the intellectual property policies of various SDOs).

[17] *See* Cargill, *supra* note 11, at 33–36.

B. *Creation Process Opportunism*

SDOs are participatory affiliations. This underpins their occasional claim to de jure status for their output, especially when operating at the consensus end of the continuum that runs from consensus processes, with their full participatory approach and decisions by voting, to consortia, which often have closed membership and less formal processes.[18] Regardless of the cooperative approach employed, an SDO's role is to facilitate agreement.

Thus, an SDO devising or revising a standard must aggregate participants' technical preferences. These ostensibly technical preferences may harbor participants' competitive strategies. This preference aggregation process is subject to public choice theory's litany of critiques, such as the influence of agenda-setting power, decisional constraints, incentives to shirk participation, factional interests, and vote trading. This list relates to opportunism concerns at the core of antitrust law: collusion among participants toward anticompetitive ends, and exclusion of participants in the standard or the market opportunities created by it thereafter.[19]

These limits to cooperation, or this detrimental cooperation, might manifest with any type of technology standard, such as standards for conduit that holds electrical wiring[20] or for underground storage tanks.[21] Other categories of creation-process opportunism discussed below, however, are exacerbated with Software Standards due to software's ability (1) to embody several types of intangible rights simultaneously; and (2) to deploy information-processing functionality into commerce, yet keep secret the methods directing such processing.

Recently, the "patent ambush" problem has attracted much attention as a specific type of intellectual property rights-clearing concern for SDO standards. In the prototypical scenario, a conniving SDO participant guides a standard toward technology for which it has, or might soon have, patent rights. Ideally for the conniver, the standard is approved before the patent or application becomes public. Thereafter, the patent provides the possibility of extracting royalty payments

[18] *See* discussion of consensus versus consortia SDOs, *supra* note 11, para. 2.

[19] ANTITRUST HANDBOOK, *supra* note 16, at 23.

[20] *See* Allied Tube & Conduit Corp. v. Indian Head, Inc., 486 U.S. 492, 496–97 (1988) (discussing steel conduit manufacturers that packed meeting to vote down approval of plastic conduit).

[21] *See* Sessions Tank Liners, Inc. v. Joor Mfg., Inc., 17 F.3d 295, 296–97 (9th Cir. 1994) (describing underground storage tank manufacturer that influenced new fire code standards, essentially making tank liners noncompliant).

under the threat of injunction. Unless some defense such as unclean hands is available, this chills development of the standard or makes it more expensive to implement. Due to the expansion of patentability in the last decade to cover software in virtually all its forms or applications, and the flood of dubious patents that followed, the patent ambush problem is as likely to crop up with Software Standards as for any other technology.

In contrast to the patent ambush case is a scenario where an SDO obtains all rights for all participant-controlled intellectual property covering the standard, including patent rights or copyright in contributed software or other textual materials. Textual materials included with the standard need intellectual property licenses of appropriate scope so that implementers can obtain and use these materials when creating products that embody the standard. If these textual materials include code examples, a firm's example-inspired implementation of the standard—even if held private via object code distribution—may be substantially similar to the example. Holding object code private, perhaps as a trade secret, points to the last category of creation-process opportunism.

SDO participants often benefit from their private information related to a standard. This creates incentives to attempt to influence the standard in directions favorable to the participant.[22] Assume that a participant has the lowest-cost manufacturing process to make conduits from a particular material at standard-specified hardness, environmental tolerances, or uniformity of dimensions. It would prefer that the standard specify this material. Similarly, an information technology standard might favor software technologies that give certain participants a competitive advantage in the post-creation phase.

C. *Implementation Opportunism*

One post-creation issue applicable to almost all standards is the possibility of some degree of de jure enactment to force the standard on an industry.[23] This benefits any groups or firms that have production advantages in implementation, control the standard, or lead the market in products for the standard. Like creation-process opportun-

---

[22] *See* Robert Axelrod, *Coalition Formation in Standard-Setting Alliances*, *in* THE COMPLEXITY OF COOPERATION: AGENT-BASED MODELS OF COMPETITION AND COLLABORATION 96, 99–101 (Robert Axelrod ed., 1997).

[23] *See* Lawrence A. Cunningham, *Private Standards in Public Law: Copyright, Lawmaking and the Case of Accounting*, 104 MICH. L. REV. 291, 293–94, 330–38 (2005).

ism, however, implementation opportunism also has scenarios exacerbating opportunism for Software Standards. The remainder of this Part highlights these scenarios using two general labels: over-implementation and under-implementation.

The less charitable phrase for over-implementation is "embrace, extend, extinguish." In this gambit, a firm adopts a Software Standard, embracing it, but adds capability to the standard, extending it. It keeps that additional capability private using object code, and hopes that its version of the implementation will become favored and widespread in the market. This could occur due to technical advantages, network effects if the firm is a successful first mover on the standard, or a firm's preexisting market prevalence.[24] A victorious gambit would extinguish the original standard.

Under-implementation relates to latent software product interoperability needs. Often, software is sold purportedly in compliance with a variety of standards, but the typical installation will not need the full interoperability derived from all the Software Standards embodied in the software. It is a general truism that most deployed software, whether in product- or custom-developed form, contains many branches of infrequently used functionality. This is in contrast to standards centering on properties of physical phenomena. These can often be more exact than data-processing standards, and are more verifiable by customers before purchasing a product. Interoperability is more difficult to evaluate and achieve with interacting software due to the variety of computing platforms, languages, and protocols involved in the information-technology ecosystem. This makes verification of the standard implementation more difficult for Software Standards. Partial or incomplete implementations are not uncommon, and sometimes are allowed by the standard as a form of technological compromise. Thus, under-implementation gives lip service to the standard without supporting it well.

[24] *See* Sun Microsystems, Inc. v. Microsoft Corp., 188 F.3d 1115, 1118–19 (9th Cir. 1999) (alleging a less than fully compliant implementation of Sun's Java technology); Commission Decision COMP/C-3/37.792, 2004 O.J. (C 290) ¶¶ 251–258, at 72–73, *available at* http://europa.eu.int/comm/competition/antitrust/cases/decisions/37792/en.pdf (finding, in Microsoft case, that a protocol to authenticate users for purposes of granting access to computing resources had been extended beyond the relevant standard); Mark A. Lemley & David McGowan, *Legal Implications of Network Economic Effects*, 86 CAL. L. REV. 479, 485–87, 515–19 (1998) (relating network economic effects to standard setting).

II. Opportunism Inhibiting Norms in Standards Development and Free and Open Source Software

Because technology standards provide public and private benefits, they are susceptible to opportunistic behavior both during creation and later during product implementation. One policy response is norms and a system of enforcement. This Part discusses norm enforcement employed to inhibit SDO participant opportunism. It focuses on the SDOAA's requirements that must be met for its antitrust law protection to attach, an enforcement mechanism for norms that helps inhibit opportunism within an SDO. Thereafter, this Part reviews FOSS norms that perform a similar function.

A. *The Backdrop of Competitor Collaboration*

The SDOAA's protection descends from two sources. First, the general tension between the procompetitive and anticompetitive effects of competitor collaboration raised the perceived need to facilitate restrained forms of such collaboration.[25] Second, as a specific example of the first source, antitrust law's special status for joint ventures preceded the SDOAA and provided a framework for it.

For standards, the procompetitive effects center on uniformity, quality, interoperability, safety, and other attributes that support, improve, or enable the markets related to products embodying the standard. For joint ventures, the positive effects arise from combining research or production assets, at least from the perspective of the SDOAA's precursor laws, which first de-trebled damages for registered research joint ventures and provided rule of reason treatment regardless of registration, and later extended that framework to production joint ventures.[26] Notably, the production joint venture provisions do not protect marketing or distribution, reflecting a traditional, goods-

[25] The potential benefits of competitor collaboration have long been recognized by the antitrust enforcement agencies. Relatively recently, however, the agencies promulgated the Collaborations Among Competitors Guidelines, expressing a framework to evaluate the relative merits and demerits of competitor collaborations. FTC & U.S. Dep't of Justice, Antitrust Guidelines for Collaborations Among Competitors (2000), *available at* http://www.ftc.gov/os/2000/04/ftcdojguidelines.pdf.

[26] The SDOAA amends the National Cooperative Production Amendments of 1993, Pub. L. No. 103-42, 107 Stat. 117, which amended the National Cooperative Research Act of 1984, Pub. L. No. 98-462, 98 Stat. 1815; renamed it the National Cooperative Research and Production Act of 1993; and extended its provisions to joint ventures for production. SDOAA, Pub. L. No. 108-237, §§ 101–108, 118 Stat. 663 (codified at 15 U.S.C. §§ 4301–4305 (Supp. IV 2004)).

based view of the production cycle.[27] Collaborative standards development aggregates intangible assets rather than physical production resources. The antitrust issues are similar, but the paradigm to facilitate production of a standard is different from the input combinations involved in a traditional goods-based production process.

B. *Norms for Consensus Standards: The Standards Development Organization Advancement Act*

Although industry-generated standards always have been important to the U.S. economy, the SDOAA enactment explicitly recognized the increasing importance of standards in high technology fields. But the enactment was not willing to grant de-trebled damages and rule of reason treatment for any SDO intending to develop a standard. The Act incorporated a set of normative requirements that the SDO must meet to qualify for such protection by referencing an Office of Management and Budget Circular from 1997 (the "Circular").[28] The Circular was written to direct federal agencies, as purchasers, to use consensus standards rather than promulgate their own. The Circular, however, only authorized the use of consensus standards developed under open, inclusive, and participatory processes.

Thus, to qualify under the SDOAA, an organization needs "attributes of openness, balance of interests, due process, an appeals process, and consensus."[29] When adhered to, these normative process requirements reduce opportunism when compared to an organization without such process. Nevertheless, public choice theory teaches that opportunism is still possible within the normative processes themselves.

The Act and the Circular broadly describe the SDOAA norms. An SDO qualifying for protection under the Act "plans, develops, establishes, or coordinates voluntary consensus standards using procedures that incorporate" the attributes listed above in a manner consistent with the Circular.[30] The Circular recites the first four open-process attributes without additional explanation, but provides an expanded definition for consensus. It defines consensus as:

> [G]eneral agreement, but not necessarily unanimity, [including] a process for attempting to resolve objections by interested parties, as long as all comments have been fairly con-

[27] 15 U.S.C. § 4301(b)(1) (2000).
[28] *See generally* OMB CIRCULAR, *supra* note 11.
[29] 15 U.S.C. § 4301(a)(8) (Supp. IV 2004).
[30] *Id.*

> sidered, each objector is advised of the disposition of his or her objection(s) and the reasons why, and the consensus body members are given an opportunity to change their votes after reviewing the comments.[31]

The SDOAA seeks generally to inhibit opportunism under the self-evident theory that an open process enables participants to check each other, each having available the threat of involving antitrust authorities or antitrust law against the others. Enforcement of the SDOAA's norms is inherent in its scheme, and relies on the desirability of de-trebled damages and the Act's other protections for the SDO.

Several aspects of the Act's language, operation, and use are relevant to note. In the SDOAA's findings section, Congress provided six points of additional context for the open-process attributes given in the Circular: (1) notice to affected parties, (2) opportunity to participate, (3) non-domination by any group through balanced interests, (4) access to information, (5) substantial agreement required for all material points after airing all views and objections, and (6) voice rights followed by consideration and appeal.[32] Such open processes may diminish a participant firm or interest group's ability to skew the standard toward its preferences. These process requirements, however, do not necessarily lead to effective rights-clearing methods for intangible rights.

Under SDOAA norms, factions are better enabled to counterbalance attempted standard-skewing, and their use helps to clear intangible rights. The more direct approach, however, is to incorporate the following processes into the SDO: (1) disclosure of such rights; and/or (2) allocation and licensing of the rights, once known. The SDOAA alludes to this goal.[33] Its definition of "standards development activity" includes "actions relating to the intellectual property policies" of the SDO.[34] This could include intellectual property disclosure requirements, but the Act itself does not specify. Rather, a House report suggests the intent to promote such disclosure.[35] Dis-

---

[31] OMB Circular, *supra* note 11, § 4(a)(1)(v).

[32] SDOAA § 102(5)(A)–(F).

[33] *Report and Recommendations on H.R. 1086 Standards Development Organization Advancement Act*, 2003 A.B.A. Sec. Antitrust L. Rep. 17–18, *available at* http://www.abanet.org/antitrust/at-comments/2003/11-03/ncrpa.pdf.

[34] 15 U.S.C. § 4301(a)(7).

[35] *See* H.R. Rep. No. 108-125, pt. 2, at 1 (2003), *as reprinted in* 2004 U.S.C.C.A.N. 651, 651 (amending prior version of report and stating, "The Act seeks to encourage disclosure

cussing licensing in the SDO, however, creates a potential dilemma: are licensing terms an intellectual property policy, or does such discussion constitute the prohibited exchange of information "relating to cost, sales, profitability, prices, marketing, or distribution of any product, process, or service that is not reasonably required for the purpose of developing or promulgating a voluntary consensus standard"?[36] The Act gives little guidance on how to resolve this tension.

With its oblique-at-best support for intellectual property rights-clearing processes, the SDOAA does not presage any particular method to unearth and clear rights. Nor does it provide clear protection for SDOs that discuss and set licensing terms. Enforcement for abuses with intangible rights still relies on antitrust law, but also depends on other law bearing on the SDO's defined processes for such rights, including contract law, state unfair competition law, and other areas.

The SDOAA provides broad norms for general behavior and vague protection for actions supported by intellectual property rights, all enforced under antitrust law. A longer treatment than this Article would discuss one of the other overarching norm examples, such as the American National Standard Institute's SDO accreditation requirements, which provide greater detail for handling intellectual property rights.[37] The SDOAA example, however, has a unique status due to its location in antitrust law. But the greater point is the presence of norms, and related enforcement mechanisms, to limit opportunism.[38]

---

by intellectual property rights owners of relevant intellectual property rights and proposed licensing terms." (emphasis omitted)).

[36] *See* 15 U.S.C. § 4301(c)(1) (Supp. IV 2004).

[37] *See* Am. Nat'l Standards Inst., Essential Requirements: Due Process Requirements for American National Standards § 3.1, at 9 (2006) (describing requirement for royalty-free or RAND licensing of any patents included in a standard).

[38] The discussion of SDOAA norms raises a question beyond the scope of this Article: would a FOSS group developing software qualify for the SDOAA's protection? FOSS development projects are not homogenous, so a case-by-base approach would be necessary to evaluate the query. Under a plain meaning of the term, software development does not seem like "standards development." The traditional conception of a technical standard might include software interoperability information, but does not include operable software code, except for the special case of a software product becoming a de facto standard. Standards in the information technology sector sometimes come with reference implementations, such as example software showing one approach to using the standard. But this is not always the case.

FOSS development also falters, however, compared to the SDOAA's open process attributes, in that it does not necessarily guarantee due process, an appeals process, or voting in the sense meant by the Circular's definition of consensus. Notice, opportunity to par-

C. *Norms in FOSS Development*

Enforcing FOSS norms starts with the FOSS license requirements for available source code, royalty-free use, and other provisions designed to perpetuate these two ideals.[39] But a variety of other norms supports FOSS, including norms that engender source code contributions and collaboration to generate the software. The force of a license is not directly behind these other norms, except perhaps for the influence of the "right to fork" inherent in the license structure, which allows a dissatisfied subgroup of the original group to take its own path with the software. Rather, contribution and collaboration norms depend on a potent ideology and, increasingly, strategic factors in computing.

Given the overlapping approaches shared by SDOs and FOSS for clearing intellectual property rights, it is worth noting that FOSS development shares many of the open-process attributes referenced by the SDOAA.[40] These attributes are inherent in the licensing scheme. The code is available and everyone is working with the code, even users who contribute only by extending the user base.[41]

Enforcement of the norms embodied in a FOSS license flows from its adoption by a software project.[42] By its terms, the license's

---

ticipate, and access to information are all high with most FOSS development, but an inner circle or founding group of developers often has primary clout for key design decisions.

[39] *See supra* note 2.

[40] *See* Cargill, *supra* note 11, at 33–36; *see also Standards-Setting and United States Competitiveness: Hearing Before the Subcomm. on Env't, Tech. and Standards of the H. Science Comm.*, 107th Cong. 3 (2001) (mentioning, in the context of a discussion of standards in the global economy, FOSS as a factor influencing the trend toward vendor-neutral standards).

[41] STEVEN WEBER, THE SUCCESS OF OPEN SOURCE 153–54 (2004).

[42] In its history, FOSS has several examples of public license evaluation mechanisms. At the time of this writing, a particularly prominent process is just underway: the GPL revision. One way to understand the GPL revision process is that it attempts to take a de facto standard—GPL Version 2 dating from 1991—and adorn it with de jure or consensus status through the public input process associated with discussion drafts issued during the process. *See* Free Software Foundation, Inc., GNU General Public License (Discussion Draft 1 of Version 3, 2006), http://gplv3.fsf.org/gpl-draft-2006-01-16.html; Free Software Foundation, Inc., GPLv3 First Discussion Draft Rationale 1–2, 5, 21 (2006), http://gplv3.fsf.org/gpl-rationale-2006-01-16.pdf. A series of discussion drafts was released to generate comments on proposed modifications to the GPL.

This revision process, however, is similar to FOSS software-development processes evaluated against the SDOAA in *supra* note 38: notice, information, and participatory opportunities are high, but the ultimate decision is not necessarily the result of voting or consensus. *See* Free Software Foundation, Inc., GPLv3 Process Definition, at iii–iv, 3, http://gplv3.fsf.org/process-definition (follow "Download as a PDF" hyperlink) (last visited Oct. 12, 2006) (stating that although submitted comments will be reviewed and associated with identified issues, substantive revisions arising from the comments are at the discretion of the revisers).

provisions persist through successive generations of the software. Another mode of enforcement is also in play: several FOSS license authors, including the GPL author, actively assert copyright in the license. Although a copyright in such works might be thin, the assertion itself may facilitate uniformity enforcement for the license by inhibiting close variants.

This Part's norm enforcement discussion centers on the license in FOSS and on antitrust law under the SDOAA for SDOs, but these ideas converge for Software Standards where the FOSS approach suggests a substitute mechanism to inhibit opportunism. It requires source code disclosure and royalty-free use, which are analogous to boundary manipulation to inhibit opportunism in historical semicommons property regimes. The next Part elaborates.

III. Standard Output Structure, Technological Boundaries, and Opportunism

As demonstrated in Part I, standard setting and implementation raise conflicting incentives. Participants who can skew a standard to their favor may seek private gain at the expense of the common benefits of greater standardization, or participants may merely position themselves for the most advantageous commercial opportunities provided by the standard. A favorable position might come from competitive advantages with standard-mandated technology, or the potential to charge for or leverage the intellectual property rights in the standard.

In this regard, a standard is analogous to a semicommons property regime, such as the open fields system surrounding a village in historical England. The private use occurs on unenclosed strips scattered throughout the open fields. Individual landowners have rights in the strips for agriculture, typically "owning" a small number of scattered strips. The public use is communal animal grazing on the common areas and strips. The animals, such as sheep, can both help the agriculture through manure left on the land and damage it by trampling the soil.[43] The figure in Appendix A shows a sample map of the

On the other hand, a form license offered to all to use is more like a voluntary consensus standard than the software output of a FOSS development team. Furthermore, FOSS license generation via collaborative processes may fit within the statutory language that defines "intellectual property policies" as standards-development activity. Thus, compared to FOSS development, a public input process for FOSS license revision or generation seems to have a greater chance to qualify for the SDOAA's protection.

[43] *See* Smith, *supra* note 4, at 132.

open fields system.[44] The shading represents scattered holdings among the landowners.

Recent scholarship theorizes this historical semicommons property to identify the substitutable role of norms and their compliance versus manipulation of property boundaries to abate strategic behavior counterproductive to the semicommons regime.[45] The relative benefits and costs of each alternative influence whether one predominates or a mixed regime results. Under this view, there is a structural inclination for boundary revision over monitoring for norm compliance when (1) the social benefits of opportunism abatement from new boundaries that disperse a participant's possibility for private gain at the expense of the common use outweigh the social costs of restructuring boundaries, and (2) this benefit/cost assessment is more advantageous than its norm compliance counterpart.[46]

The practical implementation of this framework is a preference for scattered strips over norm monitoring in the open field system. Scattering irregular-shaped strips owned by individual landowners makes it difficult for an owner to direct trampling damage to other owners' plots and garner all the benefits of nighttime foldage—the right to manure animals leave on the land. The alternative structure is for each landowner to own one contiguous plot rather than scattered, irregular-shaped strips. The scattered strips are a form of boundary manipulation inhibiting opportunism that seeks private benefit at the expense of the common good and other landowners. If the owner has

[44] The example figure in Appendix A is from the Shropshire Routes to Roots project, whose web site is at http://www.shropshireroots.org.uk (last visited Oct. 12, 2006). The direct link to the images describing the Open Fields of Sheinton is at http://www3.shropshire-cc.gov.uk/roots/packages/lan/lan_y05.htm (last visited Oct. 12, 2006). The figure in Appendix A is from Figure 9 on the Open Fields page, which uses "glebe terriers" (a document that lists the property held by an incumbent) to reconstruct the historical holdings. Sheinton Open Fields System 1747, at fig.9, http://www3.shropshire-cc.gov.uk/roots/images/lan_f41a.jpg (last visited Oct. 12, 2006). The author thanks his student research assistant, Nivine Zakhari, for communicating with the Routes to Roots project to secure permission to reprint this image.

The image in Appendix A is reprinted with permission from Dr. Trevor G. Hill, the copyright holder. Dr. Hill also asked that these annotations note that "these maps are based upon an original manor map—Shropshire Archives 6802—held on behalf of the Hon. H. Vane and the details from an analysis of the Survey Book Shropshire Archives 168/2-7."

[45] *See* Smith, *supra* note 4, at 131–33, 162–64.

[46] *See id.* at 138–46, 161–67. Both the value of the benefit/cost ratio and the levels of the benefit and cost will be important in determining which regime is preferred. Benefits that greatly outweigh corresponding costs may still not be achievable if structural or startup cost factors make bearing or funding the costs impracticable.

one plot with regular boundaries, she can herd sheep on paths that do not cross her land, reducing inopportune trampling, and perhaps move sheep to her land in the evening for the foldage benefits. Scattered strips thus have a different profile of costs and benefits for the owner compared to a contiguous, regularly shaped plot.[47]

Applying these concepts to Software Standards, contiguous plots are analogous to the SDO approach for standard output. Scattered plots are analogous to FOSS. The move from an SDO to FOSS is a boundary manipulation in the sense that the benefit/cost profile changes because FOSS requires royalty-free disclosure of oftentimes secret information—the source code—upon distribution of the software. This impacts the standard setters collectively and individually during both the creation process and later during implementation. To discuss the implications of manipulating the "boundary" around standards, this Part reviews the traditional output structure for standards and then contrasts that with FOSS.

A. *SDOs—Public Specifications with Oftentimes Private Implementations*

The traditional output of a process that creates a standard is communicated by a technical document, typically called a specification. There are numerous ways to taxonomize standards, but they all need their information published. One broad division for technology standards is performance versus design. For example, a performance specification for metal pipes used in household plumbing might specify the range of water pressure they must tolerate without bursting, or specify how much they expand or contract within a given temperature range. Design specifications aimed at the same goals, however, might specify the pipes' wall thickness and composition. Software Standards are a mixture of these two. Software design elements, such as data structures, parameter types, and program flow control, work with performance requirements, such as message throughput, acknowledgement and retry rates, and response time settings. Much of the standardization work in software is directed toward the goals of interoperability or data exchange.[48] Both design and performance requirements are important to interoperability. For software, they tend to merge the division illustrated by the above plumbing pipe example.

The mixed performance/design specification for a Software Standard guides implementers who program the standard in software

[47] *Id.* at 146–52.

[48] Another major goal is uniformity to facilitate software and human capital reuse.

deployed either in product or custom form. In most situations, the source code for this software will be held privately because, even if the software is productized, traditional proprietary software distribution includes only the object code, from which the source code is not easily discovered. Thus, the implementation of the standard is obscured. Direct inspection is not practical. Compliance can be tested in various technological ways, but the measure of compliance is only as good as the test, which is rarely complete.

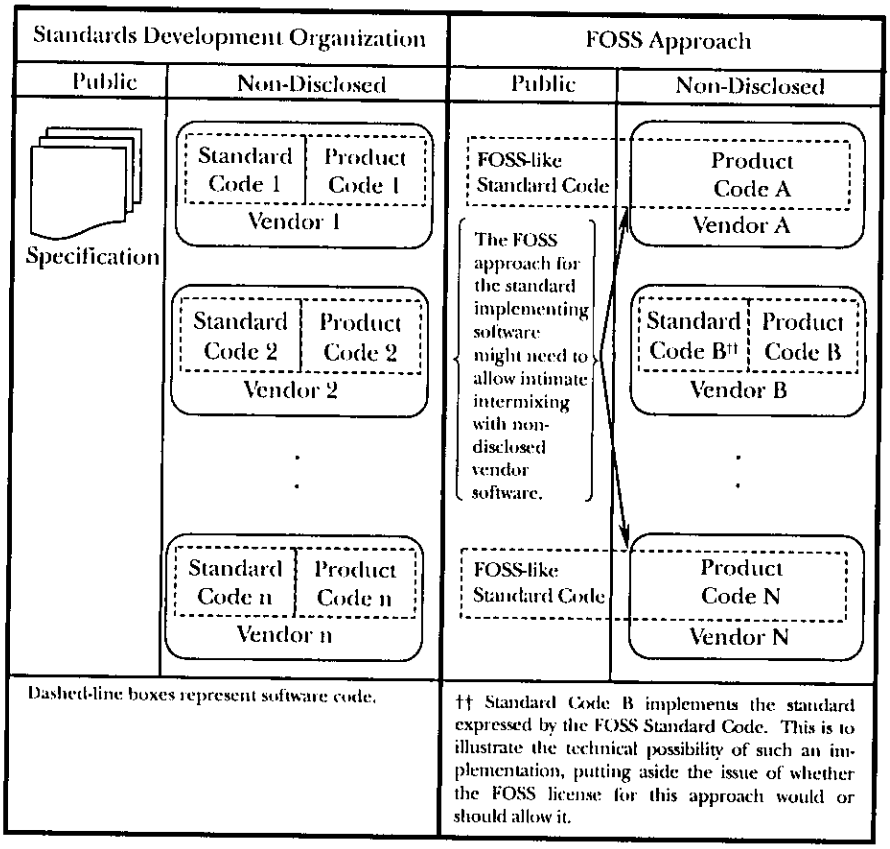
The privately held software for a Software Standard allows over-implementation and under-implementation, as those terms are used in Part I.[49] For example, a software vendor might implement a standard so that its code interoperates with the vendor's own products and third-party products. But, in this example, the vendor might invest more heavily in programming or testing compliance when interoperating with its own products, resulting in poor response times for the interface with third-party products. A user experiencing these poor response times may have difficulty pinpointing the problem or assigning causality to the correct supplier. This situation and other examples of under-implementation or over-implementation are transparent with the FOSS approach to Software Standards.

B. *FOSS—Public Implementation with Accessible Source Code*

If a standard originated as a well-documented, functioning FOSS program, the private aspects of its implementation would be greatly diminished. The source code would be available to all under the FOSS license because the software would likely be distributed. The program's comments could readily contain sufficient material to expand the inherent documentation of the standard present in the source code. In technological respects, the FOSS source code, and ancillary documents as necessary, can serve the functions of the traditional standard specification for a Software Standard, with its additional capability as operable code. Several implications result and are illustrated in the figure below.

[49] *See supra* notes 13–24 and accompanying text.

Figure—Revising the Technological Boundary for Software Standards



The FOSS approach for the standard implementing software might need to allow intimate intermixing with non-disclosed vendor software.

Dashed-line boxes represent software code.

†† Standard Code B implements the standard expressed by the FOSS Standard Code. This is to illustrate the technical possibility of such an implementation, putting aside the issue of whether the FOSS license for this approach would or should allow it.

First, presumably, the implementers would use the FOSS code in their products, provided that the FOSS license did not inhibit this. They would have to distribute or make available the source code implementing the standard, but that should be of little concern if they received it from the FOSS standard-generation process.

Second, compared to the traditional process, the FOSS approach has the potential to reduce duplicative standard implementation effort and corresponding opportunism. This presupposes that the collective efforts to generate the FOSS standard implementation are successful. Whatever collective-action or free-rider problems may exist for FOSS development generally should not be exasperated in the software-as-standard context. Even if FOSS has not fully solved these problems, the problems have not hampered FOSS so as to prohibit its emergence. FOSS standard development might more readily overcome such problems because the cooperative practices necessary to create a traditional standard will carry over in concept to FOSS and blend with its unique organizational practices.

Third, the figure illustrates the reuse of the FOSS standard software, but also shows the technical possibility of a private implementation and notes the need for the FOSS license to account for the software-as-standard. Technologically, an implementer could opt to treat the FOSS software as a technical specification document and privately implement the functionality expressing the standard. The question is whether the FOSS license should allow this. From a copyright-infringement perspective, the private implementer would need access to the FOSS software to learn the standard, and might produce a substantially similar program as to protectable elements. Some FOSS licenses view distribution of such a non-literal copy or derivative work as an action triggering the FOSS license's source code disclosure requirement. Under such a disclosure obligation, the private implementation becomes public anyway. This disincentive would likely channel firms to use the FOSS software-as-standard.

To do so, however, the software-as-standard license should consider disavowing a feature of FOSS's most popular license, the GPL. The GPL requires other closely intermixed software that might constitute a derivative work to be licensed yet again under the GPL or equivalent terms.[50] This would inhibit use of the FOSS software-as-standard because participants would often prefer to keep their product implementation secret by distributing only object code for such other software. Thus, the FOSS license for the software-as-standard should provide a safe harbor for intermixing with the implementing vendor's product software.

The final implication noted here is the political economy of FOSS as a Software Standard from the participants' perspectives. Treatment of this issue could go well beyond this Article, but the entities and employees involved in traditional information technology standard setting are not necessarily aligned with the open source faction of the FOSS movement that has taken root in some corporations. Top down, the management of an important participant in a standards process might be comfortable with the traditional standards approach, but uncomfortable with the FOSS option. Bottom up, the en-

[50] *See* Greg R. Vetter, "*Infectious" Open Source Software: Spreading Incentives or Promoting Resistance?*, 36 Rutgers L.J. 53, 129–30 (2005) (discussing the feature of the GPL oftentimes referred to as its "viral" characteristic).

gineers and technologists accustomed to working in SDOs may not be familiar with FOSS.

These issues are complicated by the diversity of software's ever-expanding domain. Whereas an Internet company might, from bottom to top, fully embrace a FOSS software-as-standard approach, an appliance manufacturer might not, even though a device as mundane as a typical new refrigerator will soon harbor significant standards-enabled software. The increasing pervasiveness of software impacts the political economy question. Information technology standard-setting work is increasingly migrating to consortia. FOSS, like the new consortia approach to standards, may also be a response to the changing nature of innovation in information technology.

In sum, rearranging the technological boundary around a Software Standard as suggested herein has a variety of implications. These implications are inherent in the move from the traditional SDO process to FOSS.[51] Using royalty-free public source code as a common implementation of the standard diminishes opportunism hidden within the object code of private implementations. The next Section of this Part reviews these effects and some limitations on their reach.

C. *Opportunism Scattering Effects and Limitations*

If participant firms migrate to FOSS for Software Standards, one explanation is that, in the aggregate, they anticipate the benefits to counter the costs in a more favorable way than the traditional SDO processes that rely in part on monitoring for norm compliance. The impetus for such a migration could be the emergence of the FOSS methodology as a cost-altering technique for aggregating creative and

[51] Movement toward the substitutability suggested by this Article can be seen through a few examples. First, some SDOs are migrating to intellectual property policies having increasing congruency with FOSS. *See* LAWRENCE ROSEN, OPEN SOURCE LICENSING: SOFTWARE FREEDOM AND INTELLECTUAL PROPERTY LAW 298–310 (2005) (discussing, in particular, the World Wide Web Consortium ("W3C") patent license and its high parallelism with FOSS licensing principles). Second, the Java programming environment, which may be a de facto standard, recently released some of its software environment as FOSS following an earlier call to do so by an influential FOSS advocate. *See* Therese Poletti, *Sun Plans to Make Some Java Code Open Source*, SAN JOSE MERCURY NEWS, June 27, 2005, at 6E; Darryl K. Taft, *Q&A: Raymond Expounds on Open Letter to Sun, McNealy*, EWEEK, Mar. 8, 2004, http://www.eweek.com/article2/0,1759,1544764,00.asp (reporting Eric Raymond's arguments that Sun should consider releasing the Java "reference implementation under an open-source license").

technological inputs.[52] Participants might readily give up remuneration for the code implementing the standard provided that they can still sell their product without disclosing its trade secrets, and the FOSS software is available to provide the standard functionality. Thus, the interface between the FOSS software-as-standard and the private product is important.

Participants could, however, attempt to shift opportunistic implementation into the product code, which might limit the beneficial effects of FOSS to generate a more uniformly implemented standard. In other words, an argument against the efficacy of FOSS for Software Standards is that it merely shifts the strategic behavior to a different layer in the technology. Whether such problems occur, or how severe they are, depends on both technological factors and the political economy of the FOSS approach in the standard-setting world.

Several issues relate to the political economy question. One is whether the FOSS approach would attract a smaller number of participants, resulting in fewer intangible rights contributed to the standard through FOSS implementation, and thus a less potent standard. The FOSS movement is engendering significant rethinking about the use of intangible rights, particularly patents, leading some major patent-holding enterprises to dedicate large numbers of patents to royalty-free use in support of FOSS and open standards. Moreover, the FOSS movement has influenced some SDOs to revise practices concerning royalties for use of a standard. The FOSS movement seeks not only royalty-free licenses, but licenses that allow free redistribution without requiring each distributor and recipient to specifically sign an agreement with the original intangible rights holder.[53] This is sometimes called "frictionless" redistribution. The explicit agreement requirement provides rights holders indirect leverage and competitive information. Each of these provisions—frictionless redistribution and royalty-free use—reduces transaction costs compared to RAND licensing that putatively requires a payment agreement with each licensee, even assuming that the parties incur no negotiating costs to decide on the reasonable royalty rate.[54] Despite whatever transaction cost advan-

[52] *See* Benkler, *supra* note 6, at 372–78; Thomas W. Merrill, *Introduction: The Demsetz Thesis and the Evolution of Property Rights*, 31 J. LEGAL STUD. 331, 334–35 (2002) (discussing the possibility of startup costs impeding the emergence of new systems of property rules).

[53] WEBER, *supra* note 41, at 85–86, 161–62, 228 (characterizing FOSS as embodying a new conception of property as a right to distribute).

[54] Concern varies over the degree of difficulty to arrive at a RAND royalty rate. *See* Miller, *supra* note 15, at 2–3, 6 (collecting sources and arguing that the RAND promise is

tages might attach to frictionless redistribution, FOSS's requirement for that novel approach could limit firms' taste for the FOSS approach to Software Standards.

These issues, however, should not obscure the overarching benefits of public source code embodying the standard. This transparency inhibits over- and under-implementation of the standard. During standard production, participants may limit strategic behavior anticipating this future transparency. Even with this benefit, however, the FOSS approach to Software Standards can do no better than SDOs for undiscovered or undisclosed third-party patent rights covering the standard. But the FOSS approach's structural characteristics suggest that it might deemphasize path dependency constraints on standardizing technology and evolving those standards.[55]

This potential result is because the FOSS license allows anyone to "fork" the implementation—to start independently down a different path with the Software Standard. The right to fork can cut both ways. It helps inhibit opportunism through the threat of exit from the FOSS development group, and can enable the standard to escape more easily a suboptimal equilibrium in a technological sense. But forking raises a fragmentation risk for the standard-implementing software, although a similar risk is present with traditional standards through over-implementation to "embrace, extend, and extinguish" the standard. One difference between the FOSS and SDO systems is the FOSS approach of free, or "frictionless," redistribution, which means royalty-free with a decentralized permission to use. Along with source code availability, these are the cornerstone characteristics of FOSS and the reasons why it proposes an intriguing alternative for Software Standards.

Conclusion

All standards face the possibility of counterproductive strategic behavior. Standards expressed in functioning software face unique forms of opportunism from implementations held private through object code. Traditional standard-development organizations often implement processes to monitor and enforce norm compliance to produce balanced, open standards. These processes can include methods to disgorge and clear intangible rights, such as participants' patent rights. FOSS norms equivalently operate through its unique licensing

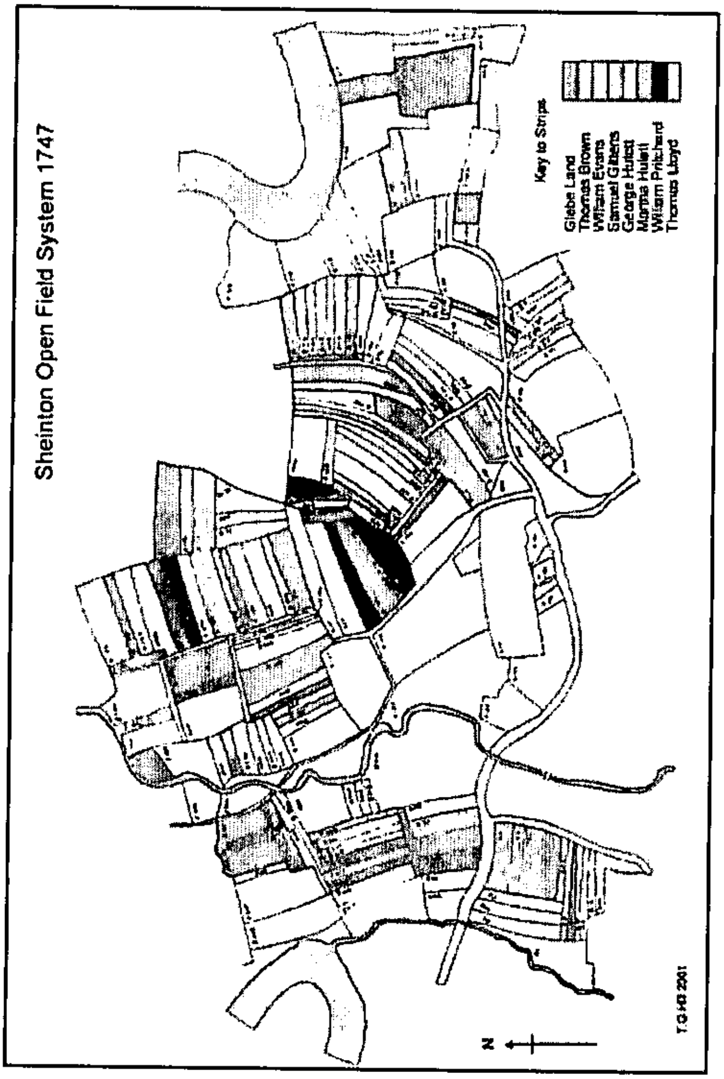
---

not so underspecified as to be inadministrable, but that it provides a guarantee of access critical to allowing coalescence around a technology).

[55] Weber, *supra* note 41, at 235–41.

structure, requiring royalty-free software with source code through successive generations of the code. For standards implemented in software, FOSS offers an alternative approach for standard generation that bypasses the traditional specification document in favor of functioning code. Various advantages and disadvantages result, including a transparent implementation with opportunism-inhibiting effects from the available source code and the characteristics of FOSS licensing.

APPENDIX A


Sheinton Open Field System 1747
Key to Strips
Glebe Land
Thomas Brown
William Evans
Samuel Gittens
George Hulett
Martha Hulett
William Pritchard
Thomas Lloyd
N
T.G.H.S 2001