HIDDEN IN PLAIN SIGHT

Michael Risch[†]

ABSTRACT

Software developers have long tried to appropriate the value of visible features and functions of their programs. Historically, they used copyright to do so, but then shifted to patenting as copyright protection for methods of operation waned. Given newfound difficulties patenting software functions, developers have one more place to turn: trade secrets.

Trade secrets have always protected the hidden aspects of programs, but can they be used to protect visible or easily discernible program features? This Article suggests a new way developers might use trade secrets to protect visible program features. It examines how they might do so, relying on traditional case law and confidentiality precautions used to keep secrets. In doing so, the Article considers whether and how such protection could be achieved in theory and in practice.

The Article then asks how software licensing would change if trade secret protection of discernible features were achieved. It considers how software might be delivered (including software as a service), the potential for trade secret trolls, the role of open source development, and the potential effect on innovation incentives.

Finally, the Article considers the alternative normative views associated with this new type of software protection.

DOI: https://dx.doi.org/10.15779/Z38MG7FV7Z

† Professor of Law, Villanova University Charles Widger School of Law. The author thanks Camilla Hrdy, Michael Madison, Sharon Sandeen as well as Jim Pooley and other co–presenters and participants of the 20th Annual BCLT/BTLJ Symposium on Software IP for their helpful comments. The author thanks Amanda Garger for outstanding research assistance.

TABLE OF CONTENTS

I. INTRODUCTION

Information wants to be free, but companies that invest money in research and development would corral it like a wild stallion. As a result, companies use patent law to protect inventions, copyright law to protect

expression, and trade secret law to protect nonpublic information. Each of these legal frameworks allow developers to extract value from information that would otherwise be freely copied by competitors. Computer software developers can easily appropriate value in the hidden parts of their products: trade secret and copyright law generally protect source code. Developers might even patent such functions if they fear independent development, but they might not know if others have also implemented a patented hidden function.

Appropriating visible features—the user interface and program operation observable by any user—has proven far more difficult. Copyright was once the developer's tool of choice to bar others from reproducing the look and feel of their product. Copyright assertion waned, however, as menu and other program element standardization took hold and courts recognized that functional elements should not receive the same protection as creative elements. The last flurry of impactful copyright cases ended in 1994 and 1996 with rulings in Apple v. Microsoft[1] and Lotus v. Borland,[2] both of which allowed for reuse of self–revealing programmatic elements.

Perhaps not coincidentally, protection for visible features shifted to patents, which steadily grew in number throughout the 1990s. These patents were spurred on by important court rulings in *In re Alappat*[3] and *State Street Bank*.[4] But as with copyrights, patenting software has lost some of its potency over time. Obtaining and asserting a software patent today is far more difficult than it was even three years ago.[5]

Enter trade secrets: left without the ability to appropriate functional characteristics by copyright and patent, software developers must look to the only form of intellectual property that will reliably protect algorithmic function. But there is a catch. As explored further below, whereas trade secrets have always been used to protect the hidden parts of software, most assume that they cannot protect the revealed aspects of software.

Those assumptions are wrong and this Article explains why. It highlights a common misconception about trade secret law: that trade secrets only cover truly secret information. For better or worse, courts have long protected information that has been disclosed to the public, sometimes

1. Apple Computer, Inc. v. Microsoft Corp., 35 F.3d 1435 (9th Cit. 1994), *cert. denied*, 513 U.S. 1184 (1995).

2. Lotus Dev. Corp. v. Borland Int'l, Inc., 516 U.S. 233 (1996).

3. 33 F.3d 1526 (Fed. Cir. 1994).

4. State St. Bank & Tr. Co. v. Signature Fin. Grp., 149 F.3d 1368 (Fed. Cir. 1998).

5. *See* Jasper L. Tran, *Two Years After* Alice v. CLS Bank, 98 J. PAT. & TRADEMARK OFF. SOC'Y 1, 3 (2016) (describing invalidations of software patents).

going as far as protecting information disclosed widely to the public, or disclosed without contract limitations. The licensing structure of software transactions provides the perfect platform for the protection of secret—but not overly so—information about product features and operations.

This examination of wide open secrets shows the continued vitality of Arrow's information paradox[6] at a time when most give it little attention. The paradox supposes that the seller must keep information secret in order to maintain its value, but the buyer cannot know the information's value without seeing it, thus destroying the secrecy. The usual solution to the paradox involves patents or non–disclosure agreements.[7] But where patents are unavailable and non–disclosure agreements are not standard operating procedure, companies attempting to protect secret information are in a real bind when they want to sell products that display that information to the general public. If companies turn to trade secrecy to protect visible program aspects, it may well change the way computer software is written, sold, and delivered.

Part II of this Article surveys the brief history of intellectual property protection, showing how copyright and patent protection leave significant gaps in protection of functionality. Part III chronicles the rise of trade secret protection to fill that gap; it may be the only intellectual property distinctly protecting algorithmic improvements. Part III then considers whether trade secrets can protect information that is viewable by users of the software. Part IV then explores the role of contracts in creating "secrecy" in revealed features, discussing doctrinal developments in favor of and contrary to this approach. Parts V and VI discuss the legal implications and innovation impacts associated with trade secret protection of revealed features. Part VII concludes.

II. A (VERY) BRIEF HISTORY OF SOFTWARE PROTECTION

Protecting computer software with intellectual property invokes a long, tortuous history that will be briefly summarized here. In the beginning, there

---

6. Kenneth J. Arrow, *Economic Welfare and the Allocation of Resources for Invention*, *in* THE RATE AND DIRECTION OF INVENTIVE ACTIVITY: ECONOMIC AND SOCIAL FACTORS 609, 615 (1962).

7. *See, e.g.*, Mattia Bianchi et al., *Selling Technological Knowledge: Managing the Complexities of Technology Transactions*, 54 RES. TECH. MGMT. 18, 24 (2011) ("The first risk that must be addressed is that of idea expropriation, which arises as soon as the firm discloses initial information about the technology. In all of the deals that we studied, the first contact with a potential partner was preceded by the signing of a non-disclosure agreement . . . .").

was only trade secret protection.[8] Even in the days of punch cards, improper access or use by another was actionable.[9] But this protection proved insufficient; it required both secrecy and wrongful appropriation or use. When a program was released into the wild, the developer could not stop others from examining it and using all of the discernable—or, readily ascertainable[10]—features and functions.

A. COPYRIGHT

Protection gradually shifted from trade secrecy to copyright, but that protection did not come easily. It was initially unclear whether software could be subject to copyright protection at all.[11] However, Congress eventually made clear that computer software could be protected by copyright and the courts began to apply Congress's mandate. But this did not end debate. While early cases found liability if one copied the entire source code of a program,[12] things grew more complicated when companies attempted to protect user interfaces—especially in cases involving non–literal copying of such interfaces.[13]

In general, user interfaces could be protected as a whole, even if their constituent parts were unoriginal.[14] An early exemplar of this approach is *Whelan Associates, Inc. v. Jaslow Dental Laboratory, Inc.*, in which the Third Circuit held that the only unprotected "idea" of a computer program is the program's purpose, and that the program constituted protectable

8. *See* David A. Rice, *Whither (No Longer Whether) Software Copyright*, 16 RUTGERS COMPUTER & TECH. L.J. 341, 342 (1990).

9. Telex Corp. v. Int'l Bus. Machs. Corp., 367 F. Supp. 258, 326 (N.D. Okla. 1973) (finding trade secret misappropriation for source code transferred via punch cards).

10. UNIF. TRADE SECRETS ACT § 1(4) (UNIF. LAW COMM'N 1985) [hereinafter UTSA] (defining trade secrets as information that is not "readily ascertainable").

11. Pamela Samuelson, *Reflections on the State of American Software Copyright Law and the Perils of Teaching It*, 13 COLUM.-VLA J.L. & ARTS 61, 61 (1988) ("[A]lmost all of the important questions about what copyright protection means for software have yet to be answered definitively."); C. Frederick Koenig III, *Software Copyright: The Conflict Within CONTU*, 27 BULL. COPYRIGHT SOC'Y U.S.A. 340, 341 (1979) ("Unfortunately just what the status of that law was on December 31, 1977, is extremely vague because of the total absence of statutory guidance in the Act of 1909 and the dearth of relevant cases dealing with this subject matter."); *see also* Atari Games Corp. v. Oman, 888 F.2d 878 (D.C. Cir. 1989) (reversing copyright office rejection of software based on lack of clarity in the rules applied).

12. Apple Comput., Inc. v. Franklin Comput. Corp., 714 F.2d 1240 (3d Cir. 1983).

13. *See* Jack Russo & Jamie Nafziger, *Software "Look and Feel" Protection in the 1990s*, 15 HASTINGS COMM. & ENT. L.J. 571 (1993) (discussing the definition and early history of "look and feel" protection).

14. Atari Games, 979 F.2d at 245–46 (holding that "breakout" game composed of geometric shapes may be protected by copyright).

expression when taken as a whole.[15] *Whelan*'s precedent has since been criticized thoroughly by many courts and commentators.[16] Despite criticism of the announced rule, a close reading of *Whelan* shows that the court was attempting to balance the incentives given to software authors and the ability to create in the future.[17]

No such concerns surfaced in *Digital Communications Associates, Inc. v. Softklone Distributing Corp.*,[18] which involved a text based command screen for a communications program. The accused command screen looked almost exactly like the plaintiff's work despite a very simple and arguably unoriginal layout. While the district court could have easily held that the command names were not copyrightable, it instead provided protection to the work as a whole and found infringement due to slavish copying.[19] These types of holdings were commonplace,[20] though not universal.[21]

The uncertainty—especially around those cases granting broad protection—came to a screeching halt in the mid–1990s. The first blow was in *Apple Computer, Inc. v. Microsoft Corp.*,[22] where the Ninth Circuit held that Apple could not protect functional elements in its desktop interface (like Apple's trash can) from being reused in functionally similar, but aesthetically different uses by Microsoft (thus, a trash can would infringe

15. 797 F.2d 1222, 1236 (3d Cir. 1986).

16. Samuelson, *supra* note 11, at 63 ("Overbroad decisions, such as that of the Third Circuit in the *Whelan* case, have set off new rounds of litigation . . . ."); *see also* Jack E. Brown, *"Analytical Dissection" of Copyrighted Computer Software–Complicating the Simple and Confounding the Complex*, 25 ARIZ. ST. L.J. 801, 814 (1993) (criticizing *Whelan*).

17. 797 F.2d at 1235 ("[W]e must remember that the purpose of the copyright law is to create the most efficient and productive balance between protection (incentive) and dissemination of information, to promote learning, culture and development."); *see also* Michael Risch, *How Can* Whelan v. Jaslow *and* Lotus v. Borland *Both Be Right? Reexamining the Economics of Computer Software Reuse*, 17 J. MARSHALL J. INFO. TECH. & PRIVACY L. 511, 516 (1999) (arguing same).

18. 659 F. Supp. 449, 460 (N.D. Ga. 1987).

19. *Id.*

20. *See, e.g.*, Lotus Dev. Corp. v. Paperback Software, Int'l., 740 F. Supp. 37 (D. Mass. 1990) (protecting structure of Lotus 1–2–3 command menu); Broderbund Software, Inc. v. Unison World, 648 F. Supp. 1127 (N.D. Cal. 1986) (protecting simple menu screen in greeting card program).

21. *See, e.g.*, Plains Cotton Co-op. v. Goodpasture Comput. Serv., 807 F.2d 1256 (5th Cir. 1987) (holding that functional constraints of cotton industry dictated similarities); Data E. USA, Inc. v. Epyx, Inc., 862 F.2d 204 (9th Cir. 1988) (holding that similarities between karate games were based on the idea of sport rather than expression).

22. 35 F.3d 1435, 1442 (9th Cir. 1994), *cert. denied*, 513 U.S. 1184 (1995).

but a recycle bin would not).[23] The court left open whether it would allow wholesale copying of an interface, but such reuse is rare.

The last influential case of that time came in 1996, when the First Circuit ruled that the Lotus 1–2–3 command structure was uncopyrightable as a method of operation.[24] The court likened the Lotus system to buttons on a VCR, and ruled that even literal copying was acceptable given the system's functionality. In so ruling, the court overruled prior cases like *Lotus Development Corp. v. Paperback Software, International*, which granted protection to Lotus 1–2–3 against a clone spreadsheet software.[25]

At about this same time Microsoft released Windows 95, which brought significantly more standardization of menu structures for software. This standardization, combined with cases like *Apple v. Microsoft* and *Lotus v. Borland,* means that there have been very few copyright cases relating to a program's look and feel—basic visible software operation. The transition was not instant, of course. Some cases continued to protect operability interfaces.[26] Nonetheless, it is telling that after the Supreme Court affirmed *Lotus v. Borland* by an equally divided court, the issue has never been squarely presented to the Court again in the twenty years since.[27]

B. PATENT

Whether coincidentally or not, the mid–1990s marked a time of tremendous growth in software patents.[28] While there is no single reason for this growth, the historical narrative allows for a coherent, if not provable, story. Leading up to the nineties, a triad of Supreme Court cases in the 1970s left uncertainty about what software could be patented. On the one hand, *Gottschalk v. Benson*[29] and *Parker v. Flook*[30] seemed to imply that algorithms were not patentable. But *Diamond v. Diehr*[31] implied that software might be patentable if combined with some generally useful

23. *Id.* at 1438 n.4.

24. Lotus Development Corp. v. Borland International, Inc., 49 F.3d 807, 814–15 (1st Cir. 1995), *aff'd by an equally divided Court*, 516 U.S. 233 (1996).

25. 740 F. Supp. at 68.

26. *See, e.g.*, Compaq Comput. Corp. v. Procom Tech., Inc., 908 F. Supp. 1409, 1421–22 (S.D. Tex. 1995) (protecting the selection of numbers used to imply hard drive compatibility, but finding the order of those numbers to be an unprotected method of operation).

27. Some might argue that *Oracle America, Inc. v. Google Inc.*, 872 F. Supp. 2d 974 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014) came close.

28. *See* James E. Bessen, *A Generation of Software Patents*, 18 B.U. J. SCI. & TECH. L. 241, 253 (2012); *see also* Bronwyn H. Hall & Megan MacGarvie, *The Private Value of Software Patents*, 39 RES. POL'Y 994, 996 (2010).

29. 409 U.S. 63, 65 (1972).

30. Parker v. Flook, 437 U.S. 584, 589 (1978).

31. Diamond v. Diehr, 450 U.S. 175, 191 (1981).

process—in that case, manufacturing rubber. This uncertainty led to limited software patenting in the 1980s.

However, software patenting exploded in the 1990s. The early nineties marked both the introduction of a commercialized internet and the World Wide Web. This created the incentive and means for a rapid expansion in software. Software was no longer limited to floppy (or compact) discs purchased at the store; every web site offered new products or services delivered over the internet, even if that service had previously been offered by traditional methods. Furthermore, copyright protection in visible programs was dwindling. After the boom of *Lotus v. Paperback Software* came the bust of *Lotus v. Borland*.

Thus, despite the uncertainty from earlier Supreme Court opinions, applicants began filing more and more software patent applications with the hope that they would eventually be affirmed. And they were: in 1994, *In re Alappat*[32] approved these software patents by ruling that general purpose computers become, in the eyes of the law at least,[33] a new machine when programmed with a new function. This opened the door to many new machine patent claims: calculators on a computer, auctions on a computer, financial management on a computer, as if each computer running a program were some new device that had been invented.

The Federal Circuit immediately began loosening up its restrictions, recognizing that the "new machine" fiction masked what was really going on: patenting novel ways of doing things on a computer. Thus, in *In re Beauregard*,[34] the court ruled that a computer program on a machine–readable medium was sufficient for a patentable claim, when ordinarily such a program would be unpatentable as inoperable printed matter.[35] Later,

32. *In re* Alappat, 33 F.3d 1526, 1545 (Fed. Cir. 1994).

33. This is, of course, a legal fiction—one that treats software as "structure" compared to the prior art. Without this fiction, a computer programmed to do X is the same as a program to do Y, because they both have all the same parts: microprocessor, RAM, etc. Michael Risch, Response Re: Request for Comments on Functional Claiming and Software Patents, Docket No. PTO-P-2012-0052 (March 12, 2013), http://www.uspto.gov/sites/default/files/patents/law/comments/sw-f_risch_20130312.pdf ("The easiest way to solve the functional claiming problem is to reverse the rule of *Alappat*, and recognize reality: machines do not become new simply because new software is loaded onto them.").

34. *In re* Beauregard, 53 F.3d 1583, 1584 (Fed. Cir. 1995).

35. *In re* Ngai, 367 F.3d 1336, 1339 (Fed. Cir. 2004) (citing *In re* Gulack, 703 F.2d 1381, 1387 (Fed. Cir. 1983)) (explaining that words, pictures and other printed matter must have some functional relationship to the information's medium of display in order to create a "new" product).

in *AT&T Corp. v. Excel Communications, Inc.*,[36] the court held that software was patentable whether it was claimed as a machine or a process, and in *State Street Bank & Trust Co. v. Signature Financial Group, Inc.*,[37] the court held that such a process need not be directed to manufacturing, and could encompass business methods so long as they yielded useful, tangible, and concrete results.

While the conventional wisdom blames *State Street Bank* for the explosion in software patenting,[38] that blame is misplaced. Many software patents issued in the late 1990s—including the patent in *State Street Bank* itself—were in the pipeline long before that opinion was issued. If any case caused the growth in patenting, it was *Alappat*; but even *Alappat* cannot be described as the sole cause. These cases were simply responsive to what was happening at the Patent Office, not causes of it. Additionally, limited patenting in the 1970s and 1980s may have helped cause the boom in the 1990s and 2000s, because there was little prior art for examiners to draw on when patent applications began flowing in.[39]

Whatever the cause, it is undeniable that software patents proliferated during the 1990s. The desirability of this growth was vigorously disputed by scholars, practitioners, and programmers. Virtually every software engineer hates software patents, or thinks they are all obvious or otherwise defective.[40] Some scholars bemoan them.[41] Other scholars find benefits in them.[42] Some say it depends on who has them.[43]

The Supreme Court has also weighed in. When the issue of business methods patents first came to the Court in 2010, sixty–eight parties filed

---

36. AT&T Corp. v. Excel Communications, Inc., 172 F.3d 1352, 1355–56 (Fed. Cir. 1999).

37. State St. Bank & Tr. Co. v. Signature Fin. Grp., Inc., 149 F.3d 1368, 1375 (1998).

38. *See, e.g.*, Francisc Marius Keely-Domokos, Comment, State Street Bank & Trust Co. v. Signature Financial Group, Inc., 14 BERKELEY TECH. L.J. 153, 171 (1999).

39. *See* Michael Risch, *The Failure of Public Notice in Patent Prosecution*, 21 HARV. J.L. & TECH. 180, 196 (2007) (discussing limitations on searching, even into 2000s); Mark A. Lemley et al., Brief in Support of Neither Party, Bilski v. Doll, 556 U.S. 1268 (2009) (No. 08-964), http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1485043 ("The perceived inability to patent software-related inventions drove such inventions 'underground' into trade secrecy . . .").

40. *See generally* GROKLAW, http://www.groklaw.net (last visited Aug. 23, 2017).

41. *See e.g.*, Bessen, *supra* note 28.

42. *See e.g.*, Michael Noel & Mark Schankerman, *Strategic Patenting and Software Innovation*, 61 J. INDUS. ECON. 481 (2013).

43. Iain M. Cockburn & Megan J. MacGarvie, *Entry and Patenting in the Software Industry*, 57 MGMT. SCI. 915 (2011) (arguing that patents benefit those who are able to use them in cross–licensing negotiations).

amicus briefs.[44] The Court issued a relatively short opinion rejecting both extremes.[45] The useful, tangible, and concrete test of *State Street Bank* was insufficient to determine patent eligibility. However, the Court would not ban all business method or software patents. It noted that 35 U.S.C. § 100(b) defines a method as a new use for an existing machine.[46] Based on this definition, software patents make perfect sense: the software is a new use for the computer. But the Court did not stop there: it ruled that even such processes must claim more than an abstract idea.[47] The patent at issue in the case essentially claimed the abstract idea of hedging,[48] and while the court said machines were not required for patentability, it did not help that the claim mentioned no software or any new use for a machine.[49]

In the immediate aftermath of the decision in *Bilski*, neither lower courts nor the Patent Office consistently interpreted the ruling. On the one hand, it seemed to rein in business methods; on the other hand, it had little to say about software patents. As a result, such patents received a bit more scrutiny, but they continued to issue and be successfully asserted for the most part.[50]

The uncertainty disappeared three years later with the Court's decision in *Alice v. CLS Bank.*[51] In *Alice*, the patentee claimed a method of settling escrow accounts by keeping "shadow accounts" that tracked the results of accumulated transactions—sometimes one party spent more, sometimes the other party spent more. At the end of the day, the shadow accounts would be compared and reconciled, with any difference paid out of the actual escrow account. In practice, this was a relatively costly system to effectively implement, as it required high speed networks, databases, audit logs, and other software programming. It took the alleged infringer years to

44. Michael Risch, *Forward to the Past*, 2009–2010 CATO SUP. CT. REV. 333, 337. The author discloses that he co–authored one of these briefs.

45. Bilski v. Kappos, 561 U.S. 593, 130 S. Ct. 3218 (2010).

46. *Id.* at 3222.

47. *Id.* at 3225.

48. Hedging is a risk mitigation strategy. In general, a party transacts with two groups with different risk profiles that roughly balance out—when one wins, the other loses.

49. For a more detailed critique of *In re* Bilski, see Risch, *supra* note 44.

50. Kevin J. McNamee, *A View From the Trenches: Section 101 Patent Eligibility Challenges in the Post*-Bilski *Trial Courts*, N.Y. INTELL. PROP. L. ASS'N BULL. (Dec. 2013), http://www.mondaq.com/unitedstates/x/297550/Patent/A+View+From+the+Trenches+Section+101+Patent+Eligibility+Challenges+in+the+PostBilski+Trial+Courts (describing outcomes of post–*Bilski* challenges).

51. Alice Corp. v. CLS Bank Int'l, 134 S. Ct. 2347 (2014).

implement after the idea was known.[52] As claimed, however, the system was remarkably simple: use two variables to track data and then compare them at some predetermined time. One amicus brief implemented what the author claimed was an infringing implementation in seven lines of basic code.[53]

The Supreme Court issued another relatively short opinion: the escrow patent was not different in kind from the hedging patent, and was therefore an abstract idea.[54] The opinion added two important pieces to the *Bilski* puzzle. First, there was a definite, "we really mean it" tone to the opinion that implied the lower courts had not heeded its prior opinion.[55] Second, the opinion provided a little more detail about how courts should go about determining if something is an abstract idea, though the framework the Court provided is extremely flexible.

The lower courts and Patent Office have taken the message in *Alice* to heart. In some technology fields, virtually every patent application is rejected. For example, district courts have invalidated a large portion of challenged patents, and the Federal Circuit has invalidated almost all of the patents it considered on appeal.[56] The application of *Alice* has been so aggressive that many historic patents would be at risk today.[57] In short, this is not a great time for software patents, whether one disagrees with the trend or not.

---

52. Joe Mullin, *How Far Will the Supreme Court Go to Stop Patent Trolls?*, ARS TECHNICA (Mar. 31, 2014), http://arstechnica.com/tech-policy/2014/03/how-far-will-the-supreme-court-go-to-stop-patent-trolls/. ("The systems used by CLS . . . are undoubtedly complex, but the bank itself is a classic example of an idea whose time had come—it was 'decades in the making,' as CLS' lawyers explain in their brief.").

53. *Id.* This claim is a bit overstated. The patent claims "communications controllers," which would ostensibly allow "messages" to come from a third parties (and for instructions to be sent back to those third parties). The seven lines of code use keyboard input and monitor output to add to internal variables as if that were messages coming over a network. Managing such data would be more complex than seven lines of code allows. Even so, the point is made that the claim is a broad one easily implemented, though one wonders if it could be implemented in seven lines of code, why it took decades to implement.

54. *Alice*, 134 S. Ct. at 2357.

55. *Id.* ("It is enough to recognize that there is no meaningful distinction between the concept of risk hedging in *Bilski* and the concept of intermediated settlement at issue here. Both are squarely within the realm of 'abstract ideas' as we have used that term.").

56. Robert R. Sachs, *#AliceStorm: July is Smoking Hot, Hot, Hot…and Versata is Not, Not, Not*, BILSKI BLOG (Jul. 13, 2015), http://www.bilskiblog.com/blog/2015/07/alicestorm-july-is-hot-hot-hotand-versata-is-not-not-not.html.

57. Michael Risch, *Nothing is Patentable*, 67 FLA. L. REV. FORUM 45 (2015).

III. FILLING THE VOID: TRADE SECRECY

Long before copyright and patent law were viable alternatives to protect software, courts consistently relied on trade secret law to protect software. The reason is straightforward: there is no definitional uncertainty in trade secret law. Indeed, functional and intangible information have unquestionably received trade secret protection since the dawn of software. The Restatement (First) of Torts defined a trade secret as "any formula, pattern, device or compilation of information which is used in one's business, and which gives" the secret holder "an opportunity to obtain an advantage over competitors who do not know or use it."[58] This included a "method of bookkeeping or other office management."[59]

Modern legislation frames trade secrets similarly to the Restatement. The Uniform Trade Secrets Act,[60] now adopted in forty–eight states, defines a trade secret as:

> information, including a formula, pattern, compilation, program, device, method, technique, or process, that:
>
> (i) derives independent economic value, actual or potential, from not being generally known to, and not being readily ascertainable[61] by proper means by, other persons who can obtain economic value from its disclosure or use, and
>
> (ii) is the subject of efforts that are reasonable under the circumstances to maintain its secrecy.

Operation of a computer program, if it otherwise satisfies the requirements of the statute, falls squarely within the information, program, device, method, technique, or process portion of the definition. Unlike copyright, there is no exclusion for functionality. Unlike patent, there is no exclusion for abstract ideas. Instead, trade secrecy only requires that the information cannot be readily known or ascertainable, that it is subject to

58. RESTATEMENT (FIRST) OF TORTS § 757 cmt. b (AM. LAW INST. 1939).

59. *Id.*

60. UTSA § 1(4).

61. Some states, most notably California, omit the requirement that the information not be readily ascertainable from the definition. CAL. CIV. CODE § 3426.1 (2012). Instead, the fact that information is "readily ascertainable" is a defense by the purported misappropriator, but only if the misappropriator actually "ascertained" the information in a legal way. Sargent Fletcher, Inc. v. Able Corp., 3 Cal. Rptr. 3d 279, 286–87 (Cal. Ct. App. 2003); ABBA Rubber Co. v. Seaquist, 286 Cal. Rptr. 518, 529 n.9 (Cal. Ct. App. 1991). In California, one may not obtain information contrary to the statute and then claim that the information would have been readily ascertainable if only the defendant had acted properly.

reasonable efforts to maintain secrecy, and that it have economic value *because* it is not known.

There are generally two types of information associated with any product: non–revealing and revealing.[62] Non–revealing information cannot be gleaned by users who have the product in ordinary distribution.[63] Revealing information is information that can be discovered by use. This might include information that is easily visible or information that can be learned through reverse engineering.[64] Software, like most other products, includes both of these types of information. And like other products, trade secret treatment may differ for each type of information.

A. NON–REVEALING SOFTWARE

Programmatic aspects that are non–revealing and not easily reverse engineered are the most easily protected by trade secret law.[65] That has always been true. Indeed, one of the primary justifications for the patent system—at least with respect to software—is that these non–revealed programmatic elements are disclosed in the patent process rather than kept secret.[66] Thus, every software author faces a choice: use protection that requires disclosure, or use protection that allows for secrecy. Copyright poses little trouble in this equation because the Copyright Office will accept redacted submissions that protect secrecy while also granting copyright protection.[67] Patents, on the other hand, usually force software authors to choose between secrecy and disclosure. In cases where software is patentable, the developer must choose whether to disclose in exchange for

62. *See* David A. Rice, *License with Contract and Precedent: Publisher-Licensor Protection Consequences and the Rationale Offered for the Nontransferability of Licenses Under Article 2B*, 13 BERKELEY TECH. L.J. 1239, 1244–45 (1998) (describing potentially self–revealing elements of software).

63. *See* J. Jonas Anderson, *Secret Inventions*, 26 BERKELEY TECH. L.J. 917, 956 (2011) (describing products that involve non–revealing information).

64. *See id.*

65. *See, e.g.*, Integrated Cash Mgmt. Servs. v. Dig. Transactions, 920 F.2d 171, 174 (2d Cir. 1990) ("The manner in which ICM's generic utility programs interact, which is the key to the product's success, is not generally known outside of ICM. Contrary to defendants' suggestion, the non–secret nature of the individual utility programs which comprise ICM's product does not alter this conclusion.").

66. *See* WILLIAM M. LANDES & RICHARD A. POSNER, THE ECONOMIC STRUCTURE OF INTELLECTUAL PROPERTY LAW 294–333 (2003).

67. Michael Risch, *Trade Secret Law and Information Development Incentives*, *in* THE LAW AND THEORY OF TRADE SECRECY: A HANDBOOK OF CONTEMPORARY RESEARCH 152, 153 (Rochelle C. Dreyfuss & Katherine J. Strandburg eds., 2010) (discussing how trade secret law affects copyright).

exclusive rights, or to hold the secret and risk independent development.[68] The choice typically depends on the likelihood that others will figure out the secret on their own, thus negating the value of a trade secret and increasing the value of a patent.[69]

However, if software is no longer patentable subject matter, logic dictates that developers will be incentivized to hide as much program functionality as possible. That includes information which would otherwise have been disclosed in the patent process. This result does not even depend on the existence of trade secret law, per se. In a world without forced disclosure, software developers would likely still attempt to rely on secrecy to minimize free–riding effects from competitors, even if the law does not provide a remedy when others attempt to learn the secret.

It is unclear how shifting from patenting to secrecy would impact social welfare. Those opposed to software patents would likely say that patents add minimal value anyway and any infringing software is usually independently developed. Those in favor of software patents would counter that early filing and disclosure might speed up dissemination, if others paid attention to patents and were also willing to license technology. The truth likely lies somewhere in between. Many software products are independently developed, but many others are not. Indeed, there may be many software programs right now that have unduplicated functionality because that functionality is not revealed.

B. SELF–REVEALING SOFTWARE

Software visible to the user, like the user interface and feature list, is a much more difficult problem—for the developer.[70] In theory, such functionality cannot be protected because it is no longer a secret. People can see the functionality and use it. Furthermore, the software might be reverse engineered, which is an acceptable way to discover a trade secret. In short, the software seems to have lost its status as "not generally known" under the statutes offering trade secret protection.

---

68. A third choice often practiced is to disclose some information while keeping other information secret. Elisabetta Ottoz & Franco Cugno, *Patent–Secret Mix in Complex Product Firms*, 10 AM. L. & ECON. REV. 142, 143–45 (2008); BRIAN C. REID, CONFIDENTIALITY AND THE LAW 64–65 (1986).

69. For a complete discussion on the tradeoffs between patent and trade secret, see Risch, *supra* note 67.

70. Brian W. Carver, *Share and Share Alike: Understanding and Enforcing Open Source and Free Software Licenses*, 20 BERKELEY TECH. L.J. 443 (2005) (explaining that "most proprietary source code is never publicly disclosed" because software authors have an incentive to keep their code secret).

But the answer is not so simple. There have long been cases that protected products—including software—that were delivered to vendors and otherwise shown publicly. In other words, despite the fact that the features are visible to users, they have not really been "disclosed" under the law. Indeed, a jury recently awarded nearly $1 billion to a software company alleging just this: that the Eclipse medical records software features and documentation, seen by nearly 300,000 users, were secret because those users required a password.[71] In another case, a plaintiff won a nearly $75 million judgment against Caterpillar[72] based on Caterpillar's attempt to recreate a coupler that was easily visible when Caterpillar purchased it, but for which Caterpillar also had confidential drawings and test data.[73] The Court denied summary judgment because the contract between the parties contained a non–disclosure agreement that maintained secrecy of a device viewable by all.[74]

There are two common threads throughout the history of trade secret protection. First, disclosures to the public will not necessarily destroy a trade secret. Second, the question is fact–specific and depends on the relationship between the parties. Not every arrangement will lead to protection.[75]

A long line of cases—in virtually every circuit—uses these two strands to provide for the protection of trade secrets in products sold to the public. For example, in *University Computing Co. v. Lykes-Youngstown Corp.*,[76] the court affirmed trade secrecy of a program sold to a large company under a limited use and disclosure license. In *TDS Healthcare Systems v. Humana Hospital Illinois, Inc.*,[77] the court allowed a jury to determine trade secrecy where hospital staff viewed software and the plaintiff demonstrated the software at trade shows, but licensed it under a confidentiality agreement.

---

71. Epic Sys. Corp. v. Tata Consultancy Servs., No. 14-cv-748, 2015 U.S. Dist. LEXIS 176561, at 1–3 (W.D. Wis. Mar. 2, 2016) (denying summary judgment).

72. Miller UK Ltd. v. Caterpillar, Inc., No. 10-cv-3770, Dkt. No. 972 (N.D. Ill. Dec. 18, 2015).

73. Miller UK Ltd. v. Caterpillar, Inc., No. 10-cv-3770, Dkt. No. 777-2 (N.D. Ill. Sep. 8, 2015).

74. Miller UK Ltd. v. Caterpillar, Inc., No. 10-cv-3770, Dkt. No. 1, at 8–9 (N.D. Ill. June 7, 2010).

75. Stargate Software Int'l v. Rumph, 482 S.E.2d 498, 502 (Ga. Ct. App. 1997) (holding that suggesting ex–employee's work on a particular project at another company is not reasonable precaution).

76. Univ. Computing Co. v. Lykes-Youngstown Corp., 504 F.2d 518, 535 (5th Cir. 1974).

77. TDS Healthcare Sys. Corp. v. Humana Hosp. Ill., Inc., 880 F. Supp. 1572, 1578–84 (N.D. Ga. 1995).

In *Trandes Corp. v. Guy F. Atkinson Co.*,[78] the court affirmed a jury verdict of misappropriation of computer software object code based solely on an agreement that customers would use the software for internal purposes only (though the plaintiff had only sold a few copies). In *Micro Data Base Systems v. Dharma Systems, Inc.*,[79] Judge Posner affirmed a finding of trade secrecy without a promise of confidentiality and without questioning how computer software that was widely available became a trade secret when slightly modified, based on contractual provisions. In *SOS, Inc. v. Payday, Inc.*,[80] the Ninth Circuit held that software licensed without a confidentiality provision did not destroy trade secrecy if other versions of that software were obtained in breach of confidence—with or without a contractual confidentiality provision. In *Data General Corp. v. Grumman Systems Support Corp.*,[81] the court held that widely distributed software remained a trade secret because it was subject to a license agreement that required confidentiality and return upon non–use.

While those who favor the public domain may criticize these cases, they are not anomalies. Courts have long been willing to find breaches of confidentiality even where a device is on the market. For example, in *Hyde Corp. v. Huffines*,[82] the Texas Supreme Court affirmed an injunction against a breacher despite the fact that the product had been on sale at the time of the breach (and subsequently patented). In *Smith v. Dravo Corp.*,[83] the Seventh Circuit held that a publicly sold device would not preclude a trade secrecy claim if the misappropriator breached a confidence to obtain a copy: "[T]he mere fact that such lawful [public] acquisition is available does not mean that he may, through a breach of confidence, gain the information in usable form and escape the efforts of inspection and analysis."[84] Similarly, in *K-2 Ski Co. v. Head Ski Co., Inc.*,[85] the secret skis were displayed and discussed at a conference attended by materials manufacturers, but not

78. Trandes Corp. v. Guy F. Atkinson Co., 996 F.2d 655, 664 (4th Cir. 1993).

79. Micro Data Base Sys., Inc. v. Dharma Sys., Inc., 148 F.3d 649, 656–57 (7th Cir. 1998).

80. *See* S.O.S., Inc. v. Payday, Inc., 886 F.2d 1081, 1089–90 (9th Cir. 1989).

81. Data Gen. Corp. v. Grumman Sys. Support Corp., 36 F.3d 1147, 1167–70 (1st Cir. 1994).

82. Hyde Corp. v. Huffines, 314 S.W.2d 763, 777 (Tex. 1958).

83. Smith v. Dravo Corp., 203 F.2d 369, 375 (7th Cir. 1953).

84. The California implementation of the UTSA would arguably lead to a similar outcome if the unprotected version were not *too* widespread. For a further discussion, see Michael Risch, *Why Do We Have Trade Secrets?*, 11 MARQ. INTELL. PROP. L. REV. 1, 55–57 (2007).

85. K-2 Ski Co. v. Head Ski Co., Inc., 506 F.2d 471 (9th Cir. 1974).

direct competitors; the court affirmed the existence of trade secrets.[86] Finally, in *Metallurgical Industries Inc. v. Fourtek, Inc.*,[87] the court ruled that two disclosures without express confidentiality agreements did not vitiate trade secrecy because they were limited and non–public. To be sure, many of the older cases were decided under the Restatement, but the newer cases come to the same conclusion. Further, there is no particular section of the UTSA that would mandate different outcomes even if modern drafters intended to limit protection. Finally, the remedy in such cases may simply be a "head start" injunction that delays release of the features.

Of course, none of these—except perhaps for the jury verdict favoring the Eclipse medical record software[88]—were mass–market products. The discussion below considers how widespread distribution alters the analysis.

C. KEEPING THE CAT IN THE BAG: MAINTAINING SECRECY OF REVEALED FEATURES

The goal for developers would be to find some way to satisfy the statute's requirements for non–ascertainability and reasonable precautions—no small feat for program features visible to any user. There are three primary ways they might attempt this: non–disclosure agreements, anti–reverse engineering agreements, and reliance on norms.

1. Non–Disclosure Agreements

The most basic way to satisfy the statute is to obtain non–disclosure agreements from potential and actual purchasers.[89] Non–disclosure agreements are most easily obtained from actual users through a click–wrap agreement. The contractual implications of such consumer agreements are discussed below, but the agreements would not stop at the point of use. Visitors to websites would have to agree. Customers visiting a company would have to sign one. Sales materials would have to be marked confidential.

In short, the entire sales chain would be locked down, which is easier said than done, especially for software sold face–to–face by commissioned

86. *Id.* at 474–75.

87. Metallurgical Indus., Inc. v. Fourtek, Inc., 790 F.2d 1195, 1200–02 (5th Cir. 1986).

88. Epic Sys. Corp. v. Tata Consultancy Servs., No. 14-cv-748, 2015 U.S. Dist. LEXIS 176561, at 1–3 (W.D. Wis. Mar. 2, 2016) (denying summary judgment).

89. Victoria A. Cundiff, *Reasonable Measures to Protect Trade Secrets in a Digital Environment*, 49 IDEA 359, 375 (2009) ("While in the tangible world it may not always be essential to require parties receiving access to trade secrets to sign a NDA, it is good practice. In the digital world, however, it can be essential." (citation omitted)).

employees. Salespeople may have an incentive to (and have been known to)[90] disregard sales cycle confidentiality procedures and to share information with customers, if that information would help effectuate the sale. More generally, it is very difficult to ensure that every user of a product agrees to a non–disclosure agreement. Even so, secret sales chains are not unprecedented.[91] Nonetheless, such changes would likely affect how software is distributed, a topic discussed below.

2. Anti–Reverse Engineering

To limit reverse engineering, non–disclosure agreements would also include a "no reverse engineering" clause, which is relatively standard in today's license agreements. This clause should bar the customer/competitor from digging deeper to determine how the program works, making what would otherwise be proper reverse engineering into an unlawful, improper acquisition of information. This clause would best apply to non–visible program aspects that were otherwise easily discernible, and would apply regardless of whether the visible aspects constitute secrets. It therefore provides an additional layer of protection: even if the visible features are not confidential, their implementation might be.

3. Reliance on Norms

In some industries, the developer may be able to rely on norms to protect trade secrets. As noted above, many cases have upheld secrecy of nominally non–secret information when the parties involved could be expected not to use or disclose the information.[92] This is a risky method of protection, however, as the developer is betting ex ante that it knows the norms, can find an expert to testify to those norms, and will convince the fact–finder to agree. This is not a particularly strong way to protect a trade secret, but it can patch shortcomings in a contract and serve as a last resort when employees make mistakes.

90. *See, e.g.*, Data General Corp v. Digital Computer Controls Corp., 297 A.2d 437, 438 (1972); *see also* anecdotes on file with author.

91. *See* Miller UK Ltd. V. Caterpillar, Inc., No. 10-cv-3770, Dkt. No. 1, at 8-9 (N.D. Ill. June 7, 2010); further, the author represented a company that attempted to maintain such secrecy.

92. *See also* Hicklin Eng'g, L.C. v. Bartell, 439 F.3d 346, 350 (7th Cir. 2006) ("[E]very decision we could find applying that statute holds that an implied undertaking to abide by the trade's norms of confidentiality suffices," citing cases from multiple states).

IV. ENFORCEABILITY OF AGREEMENTS

This Article cannot fully analyze the propriety of non–disclosure agreements included in software agreements; books have literally been written on this topic.[93] Presumably any contract would have to fulfill the basics of online/software contracting today: notice and assent.[94] Thus, discussed below are some defenses to the application of contracts for secrecy: unconscionability, misuse, and preemption.

A. UNCONSCIONABILITY/LACK OF NOTICE

Software users might argue that a secrecy limitation is unconscionable. Unconscionability doctrine has long been used to allow parties to avoid unfair or oppressive contracts or terms. It is a construct of state contract law, and thus transcends issues of intellectual property. However, courts will often look to the underlying business purposes of a clause to determine whether it is unfair,[95] and intellectual property owners might argue that protection mandates non–disclosure clauses in contracts.

On the one hand, courts have been generally unwilling to find unconscionability in the case of limitations on use.[96] On the other hand, a requirement of secrecy—which admittedly breaks from historical norms—might be considered so surprising and unfair that courts would find it unenforceable. In practice, the outcome might depend on the defendant. Typical end users might be not be held to the contract, but, as in *Davidson* and *ProCD*, parties that obtain the software to explicitly compete may not be looked upon kindly by courts.[97]

Similar to unconscionability, courts might consider whether the inclusion of surprising, non–negotiated terms should be disregarded, as generally discussed in Restatement (Second) Contracts § 211(3), the

93. *See e.g.*, MARGARET JANE RADIN, BOILERPLATE: THE FINE PRINT, VANISHING RIGHTS, AND THE RULE OF LAW (2013).

94. Specht v. Netscape Commc'ns Corp., 306 F.3d 17, 28 (2d Cir. 2002) ("[W]e conclude that the district court properly decided the question of reasonable notice and objective manifestation of assent . . . .").

95. *See, e.g.*, CAL. CIV. CODE § 1670.5(b) ("[T]he parties shall be afforded a reasonable opportunity to present evidence as to its commercial setting, purpose, and effect to aid the court in making the determination.").

96. Davidson & Assoc. v. Internet Gateway, 334 F. Supp. 2d 1164, 1180 (E.D. Mo. 2004) (holding that limitation on use and reverse engineering did not "shock the conscience.").

97. *Id.* at 1179 ("[T]he defendants are not unwitting members of the general public as they claim. They are computer programmers and administrators familiar with the language used in the contract, and have the expertise to reverse engineer and understand source code.").

doctrine of reasonable expectations. This rule says that if a non–negotiated contract clause would be surprising and unexpected to the consumer, then it should be ignored. This rule is applied extensively in insurance cases, but not as often elsewhere.[98] Nonetheless, it is the type of contract defense that parties may attempt to apply here.

B. MISUSE

Courts have generally affirmed limited use contracts. In *ProCD*, the court affirmed a "no commercial use provision" holding that contractual relations expressly countenance limitations upon distribution.[99] In *Bowers v. Baystate*, the court affirmed verdict finding breach of a no–reverse–engineering contract.[100]

There are exceptions to the general friendliness to use limitations in contracts. In *Lasercomb*, for example, the court held that it was copyright misuse for a license agreement to forbid the licensee to make any kind of competing program.[101] But even this case is tenuous when applied outside its specific facts. First, it is a copyright case; it is unclear that there is a "trade secret misuse" doctrine, though aggrieved end users could certainly argue as much. Second, the non–compete in *Lasercomb* was very broad; a trade secret plaintiff might argue that a narrower clause against using the information in the program to compete is valid.[102] Third, despite finding copyright misuse, the court in *Lasercomb* upheld a fraud verdict, reasoning that the promise not to use Lasercomb's copyrighted software was made falsely.[103] That is, even when the intellectual property was misused, the

98. Ronald J. Gilson et al., *Text and Context: Contract Interpretation as Contract Design*, 100 CORNELL L. REV. 23, 82 (2014) ("The appeal, but also the limit, of reasonable expectations as a standalone special purpose insurance contract doctrine was its generality.").

99. ProCD, Inc. v. Zeidenberg, 86 F.3d 1447, 1454 (7th Cir. 1996) ("A law student uses the LEXIS database, containing public-domain documents, under a contract limiting the results to educational endeavors; may the student resell his access to this database to a law firm from which LEXIS seeks to collect a much higher hourly rate? [No]."); *see also Davidson & Assoc.*, 334 F. Supp. 2d at 1182 (limited use and anti–reverse engineering agreements do not constitute misuse).

100. Bowers v. Baystate Techs., Inc., 320 F.3d 1317, 1326–27 (Fed. Cir. 2003).

101. Lasercomb Am., Inc. v. Reynolds, 911 F.2d 970, 978 (4th Cir. 1990).

102. *Cf.* Serv. & Training, Inc. v. Data Gen. Corp., 963 F.2d 680, 690 (4th Cir. 1992) (distinguishing *Lasercomb*: "[A]ppellants have offered no evidence that Data General did anything beyond limiting the use of the software to repair and maintenance of specific computer hardware, activity that is protected as an exclusive right of a copyright owner.").

103. *Lasercomb*, 911 F.2d at 980; *see also Davidson & Assoc.*, 334 F. Supp. 2d at 1182–83 ("Finally, the Court is reluctant to apply the copyright misuse defense as a defense to a contract claim, because the defense is normally used in copyright infringement actions . . . ."); Pollstar v. Gigmania, Ltd., 170 F. Supp. 2d 974, 982 (E.D. Cal. 2000) ("[T]he court

contractual remedy was upheld. Thus, a trade secret claim based on breach of confidentiality agreements might also continue to apply because it relies on an underlying contract that is not subject to a misuse claim.

C. PREEMPTION

One might argue that contracts barring use of information are preempted. When a federal authority speaks to a legal issue, it will often negate any attempts by the subordinate authority to regulate the same issue. For example, with a few exceptions only federal laws can regulate patentable or copyrightable subject matter. Similarly, the trade secret statute preempts common law protection for secret information. But preemption can be complex.

The case most relied upon for this preemption of licensing agreements is *Vault v. Quaid*, which held that a contract barring decompiling and disassembling software was in conflict with important copyright policies allowing use of uncopyrightable information.[104] However, reliance on *Vault* by other courts has been sporadic at best. Most cases addressing copyright preemption have applied only the statutory preemption provision, finding that the addition of a contract is an "extra element" that leaves agreements enforceable.[105]

Vault is also a mixed bag from a trade secret preemption point of view. The district court ruled that the trade secrets claim was *not* preempted by the copyright act, but also ruled that reverse engineering was not a misappropriation because the contractual restriction *was* preempted.[106] The counterfactual of this Article is that such a contract would have not only barred reverse engineering, but would have made the entire software confidential, such that use of what was learned would be disclosure of a trade secret. Under this counterfactual, the trade secret claim in confidential software would likely not be preempted under *Vault*.

The UTSA similarly has a statutory preemption clause[107]: any law that protects the same thing as a trade secret is preempted. Courts differ on whether a contract claim based on breach of a non–disclosure agreement is

need not decide whether there was copyright misuse because Plaintiff does not allege copyright infringement.").

104. Vault Corp. v. Quaid Software Ltd., 847 F.2d 255, 269 (5th Cir. 1988).

105. *See infra* note 124.

106. Vault Corp. v. Quaid Software Ltd., 655 F. Supp. 750, 763 (E.D. La. 1987), *aff'd*, 847 F.2d 255, 268 n.26; *see also* Jeffrey T. Haley, *Trade Secrets & Computer Software*, 37 WASH. ST. BAR NEWS 51, 55 (1983) (explaining that trade secrets are not preempted by copyright law).

107. UTSA § 7.

preempted by the trade secret statute.[108] That is, in some states one cannot protect information with a non–disclosure agreement unless it rises to the level of a trade secret. But a case ruling that the very contract used to protect a secret was preempted by the trade secret statute is difficult to imagine. Indeed, contracts are used to create and protect the trade secret. They are part and parcel of reasonable precautions.

Public policy might impose independent limitations on the ability to contract for trade secret limitations. Courts may well decide that public policy requires that certain secrets may not be protected, even by contract.[109] Voting machines, environmental damage, and even visible elements of computer software may be among those things that should not qualify for trade secrecy. But the law does not currently impose any such limitations.

D. THE IRRELEVANCE OF CONTRACTS

Potential defendants might not be without recourse, even assuming that each particular contract is valid. A contract does not automatically confer trade secret status. Instead, the secret must be unknown and not readily ascertainable, as well as deriving economic value from secrecy.

Taking each of these in turn, can a widely distributed program really be considered not generally known (and not ascertainable)? Most of the cases cited above supporting trade secret status were for limited distribution software, and these issues were concerns thirty years ago.[110] Put another way, even if every licensee agrees to confidentiality, one could argue that it is not reasonable to expect that no competitor will ever get its hands on information about the software. Then again, trade secrecy for widespread information is not unprecedented.[111]

---

108. Robert Unikel, *Bridging the Trade Secret Gap: Protecting Confidential Information Not Rising to the Level of Trade Secrets*, 29 LOY. U. CHI. L.J. 841, 882 (1997) (discussing protection for information that is confidential but not a trade secret).

109. *See generally* David S. Levine, *Secrecy and Unaccountability: Trade Secrets in Our Public Infrastructure*, 59 FLA. L. REV. 135 (2007).

110. *See* Haley, *supra* note 106, at 58–59 (describing risks to keeping computer software secret).

111. *See, e.g.,* Religious Tech. Ctr. v. Netcom On-Line Com., 923 F. Supp. 1231, 1254 n.25 (N.D. Cal. 1995) ("The notion that the Church's trade secrets are disclosed to thousands of parishioners makes this a rather unusual trade secrets case. However, because parishioners are required to maintain the secrecy of the materials, the court sees no reason why the mere fact that many people have seen the information should negate the information's trade secret status."); Data Gen. Corp. v. Digital Comput. Controls, Inc., 357 A.2d. 105, 108 (Del. Ch. 1975) (finding trade secrets intact despite distribution of computer information to six thousand users).

At the very least, support forums, blogs, and similar media will often discuss software.[112] Even unauthorized widespread distribution of software would destroy the secret.[113] As a result, software owners might need to take even greater precautions to limit online discussion, a task made more difficult by restraints on the ability of aggrieved parties to obtain injunctions against undesired speech online.[114] But all this means the task is difficult, not impossible. The right software with the right protections could avoid these troubles, and aggressive software sellers might try just in case.

Secondly, the independent value must be derived from secrecy. But when software is widely distributed and the "secret"[115] features are visible to all users, does the value derive from functionality and distribution, or from secrecy? On the other hand, if the goal is keeping competitors from implementing identical features, then the value would come from keeping the features out of their hands—if that is possible. Consider the Eclipse medical record software case[116] discussed above. Some 300,000 people used both the external facing (user interface) components and the internal facing (data processing) components. Some of the value to the *seller* of the visible features must come from their widespread use. Interoperability, reduced training costs, and other benefits result from the widely distributed, quality medical record system whether or not the features were publicly known. But there may be another benefit to the seller associated with keeping those features, no matter how simple they are once known, out of the hands of competitors. Indeed, the simpler it is to implement a feature, the more valuable it may be to keep a competitor from knowing it. Otherwise, competitive systems might lose their distinctive functions. While potentially good for competition and user pricing, allowing competitors to easily and cheaply implement features that took time and money to develop is not a welcome result for companies investing money in research and development.

---

112. Cundiff, *supra* note 89, at 395 ("Some communities regard posting trade secrets on the Internet as a sport.").

113. *See* DVD Copy Control Ass'n Inc. v. Bunner, 116 Cal. App. 4th 241, 255 (2004) (denying injunction due to lack of secrecy: "The evidence in the present case is undisputed that by the time this lawsuit was filed hundreds of Web sites had posted the program, enabling untold numbers of persons to download it and to use it.").

114. *See, e.g.*, 47 U.S.C. § 230 (2012) (immunizing internet content posted by third parties); Stevo Design, Inc. v. SBR Marketing Ltd., 919 F. Supp. 2d 1112, 1127 (D. Nev. 2013) (barring trade secret claim under CDA Section 230).

115. Note, of course, that secrecy derives from contracts and practical ability to keep competitors from seeing the features.

116. Epic Sys. Corp. v. Tata Consultancy Servs., No. 14-cv-748, 2015 U.S. Dist. LEXIS 176561, at 1–3 (W.D. Wis. Mar. 2, 2016) (denying summary judgment).

Third, to the extent that state law protects visible features despite widespread (albeit nominally confidential) distribution, the risk of preemption increases, as noted by the Supreme Court[117]:

> The Florida scheme blurs this clear federal demarcation between public and private property. One of the fundamental purposes behind the Patent and Copyright Clauses of the Constitution was to promote national uniformity in the realm of intellectual property . . . . This purpose is frustrated by the Florida scheme, which renders the status of the design and utilitarian "ideas" embodied in the boat hulls it protects uncertain. Given the inherently ephemeral nature of property in ideas, and the great power such property has to cause harm to the competitive policies which underlay the federal patent laws, the demarcation of broad zones of public and private right is "the type of regulation that demands a uniform national rule."[118]

The Court made clear that as the public domain was deprived of rightful access (in the form of limits on the ability to reverse–engineer or otherwise copy public but federally unprotected features),[119] then the state law is more likely to be preempted. But preemption is not a foregone conclusion for all additional protections; the statute invalidated by the Court protected *all* boat hull designs, and not just those subject to secrecy agreements.[120] A statute protecting only secret designs would likely not be preempted.

V. IMPLICATIONS

This section considers some of the implications of protecting revealed software features. First, it considers whether such protection will improve exclusivity. Second, it addresses how the recent Federal Trade Secret law might change the argument. Third, it discusses how patenting might change, both by limiting prior art and encouraging plaintiffs who do not make a

117. Bonito Boats, Inc. v. Thunder Craft Boats, Inc., 489 U.S. 141, 155 (1989) ("[B]ecause the public awareness of a trade secret is by definition limited, the Court noted [in *Kewannee v. Bicron*] that 'the policy that matter once in the public domain must remain in the public domain is not incompatible with the existence of trade secret protection'.").

118. *Id.* at 162–63 (citations omitted).

119. *Id.* at 155 ("[In *Kewannee*, the] public at large remained free to discover and exploit the trade secret through reverse engineering of products in the public domain or by independent creation.").

120. *Id.* ("[C]ertain aspects of trade secret law operated to protect noneconomic interests outside the sphere of congressional concern in the patent laws. As the Court noted, '[A] most fundamental human right, that of privacy, is threatened when industrial espionage is condoned or is made profitable'.").

product to strategically assert claims. Fourth, it examines how trade secret protection might change software delivery.

A. WILL PROTECTING REVEALED FEATURES PROVIDE EXCLUSIVITY?

An ultimate question is whether trade secret protection of user interfaces and other revealed elements will make a difference in appropriability. To the extent that all similarities are independently created, then trade secrecy provides little additional exclusionary right because everyone is making their own product without reference to others.

There is some support for the assumption that all products are independently created, but the evidence is mixed. One study of patent complaints implies that little copying exists.[121] But that study looked for allegations of copying in the plaintiffs' complaints. Where the plaintiff has a product, it may not know whether the defendant ever looked at its product and thus could not allege that fact. More important, user interfaces and features lists are not quite like patents. Companies are generally aware of how their competitors' products operate. Their sales people know because they often compare features. Their support staff know because they get feature requests to match another product. The world knows because users will often gripe that a product lacks a certain feature that is present in another product. In a trade secret case, copying of—or at least access to—features may be more provable.

Our world is not one of pure independent development, but neither is it one of pure copying. Not all features are copies, and when they are implemented they may be implemented in different ways. And the plaintiff may not be able to tell where product ideas came from.

And where others have copied, then trade secret protection would allow one party to exclude the copiers, requiring litigation. Trade secret cases are messy, and the problem of determining whether a competitor copied is nothing new. Expanding secrecy into user interfaces would likely increase the number of cases and make them more difficult to litigate.

B. FEDERAL TRADE SECRET STATUTE

Congress recently enacted a federal civil trade secret statute. It is unclear whether this statute provides any real differences over the current state law.[122]

---

121. Christopher Anthony Cotropia & Mark A. Lemley, *Copying in Patent Law*, 87 N.C. L. REV. 1421 (2009).

122. *Compare* Christopher B. Seaman, *The Case Against Federalizing Trade Secrecy*, 101 VA. L. REV. 317 (2015), *with* James Pooley, *The Myth of the Trade Secret Troll: Why*

But that question aside, a federal statute might undermine the type of trade secret protection envisioned here. The federal statute explicitly states that it shall not preempt any other law.[123] Thus, preemption is unlikely. Nonetheless, a broad reading of the federal trade secrets law could wind up limiting state based attempts to protect trade secrets if those attempts conflict with the federal law. Protecting visible elements of computer software will require extensive reliance on contracts that limit use, disclosure, and reverse engineering of software. But a court interpreting federal trade secret law might, for example, consider the ability to reverse engineer to be an important federal policy. Whereas state law does not generally preempt contract law to the contrary, a federal policy might do so.[124] It is likely that this would never happen[125] given the provision in the statute, but the possibility is worth considering.

Additionally, the federal legislation declares that it is not an intellectual property statute.[126] As a result, federal trade secrets would not be exempt from Section 230 of the Communications Decency Act, which immunizes online content providers from lawsuits relating to material provided by others. This could lead to the same concerns about broad dissemination discussed above. Once information about software features is posted online, trade secret owners cannot police disclosures. To the extent that the drafters of the federal trade secret act wanted such policing (presumably so, given

We Need a Federal Civil Claim for Trade Secret Misappropriation, 23 GEO. MASON L. REV. (2016); *see also* Michael Risch, *Defending a Federal Trade Secrets Law*, WRITTEN DESCRIPTION (Nov. 19, 2015), http://writtendescription.blogspot.com/2015/11/defending-federal-trade-secrets-law.html.

123. Defend Trade Secrets Act § 2(f), Pub. L. No. 114-153, 130 Stat. 382 (2016) ("Nothing in the amendments made by this section shall be construed . . . to preempt any other provision of law.").

124. *See, e.g.*, Kewanee Oil Co. v. Bicron Corp., 416 US 470, 479 (1974) ("The only limitation on the States is that in regulating the area of patents and copyrights they do not conflict with the operation of the laws in this area passed by Congress . . . ."); Bowers v. Baystate Techs., Inc., 320 F.3d 1317, 1323–24 (Fed. Cir. 2003) (holding that copyright does not preempt contract); ProCD v. Zeidenberg, 86 F.3d at 1454 (holding that copyright does not preempt contract); Data Gen. Corp. v. Grumman Sys. Support Corp., 36 F.3d 1147, 1164–65 (1st Cir. 1994) (holding that misappropriation of computer program is not preempted by Copyright Act); TDS Healthcare Sys. v. Humana Hosp. Ill., Inc., 880 F. Supp. 1572, 1581 (N.D. Ga. 1995) (finding that misappropriation of computer program was not preempted by Copyright Act); Nat'l Car Rental Sys. v. Computer Assocs. Int'l, 991 F.2d 426, 432 (8th Cir. 1993) (ruling that limited use provision of contract not preempted by Copyright Act).

125. Aronson v. Quick Point Pencil Co., 440 U.S. 257, 262–63 (1979) (holding that federal policies did not preempt contract to pay for trade secrets even if patent was denied).

126. Defend Trade Secrets Act § 2(g).

that more protection is the stated purpose of the statute),[127] then this provision runs contrary to policing efforts. At the same time, state trade secret laws might still be considered "intellectual property" under Section 230, a question on which courts disagree. This is a complex and unsettled area.[128]

C. EFFECT ON PATENT PRIOR ART

Treating software as trade secrets could have an impact on patenting. To the extent that software is publicly released, it can be used as prior art. But if the software is kept secret through non–disclosure agreements, then it cannot bar a future invention by others.[129] The risk is that secret software will make it more difficult to invalidate later patents claiming broad, generic software functions. Indeed, a failure to allow patents on software in the 1970s and 1980s may have led to the explosion of patenting in the 1990s.[130]

Given the recent direction of patentable subject matter cases, a loss in the public domain may not matter. It may be difficult to obtain software patents in the future, regardless of trade secret concerns.

D. THE ROLE OF NPES

One might be tempted to say that non–practicing entities—NPEs, or those that do not make a product—would lose out in the shift from patent to trade secrecy. Because the point of trade secret protection is to keep secret information in software placed on sale, then one would think that those not selling any software would have no place.

This view may be too short sighted, however. Non–practicing entities, whether independent inventors, failed companies, or even technology buyers, might still be in the business of licensing their secret technology. But the transaction would be very different for NPEs. Rather than focusing

127. James Pooley, *What You Need to Know about the Amended Defend Trade Secrets Act*, PATENTLY-O (Jan. 31, 2016), http://patentlyo.com/patent/2016/01/amended-defend-secrets.html ("The DTSA in its current form is a strong bill, meeting its original objective of giving plaintiffs access to federal courts. . . .").

128. *See generally* Eric Goldman, *The Defend Trade Secrets Act Isn't an "Intellectual Property" Law*, 33 Santa Clara High Tech. L.J. 541 (2017).

129. *See* W.L. Gore & Assocs., Inc. v. Garlock, Inc., 721 F. 2d 1540, 1550 (Fed. Cir. 1983) ("As between a prior inventor who benefits from a process by selling its product but suppresses, conceals, or otherwise keeps the process from the public, and a later inventor who promptly files a patent application from which the public will gain a disclosure of the process, the law favors the latter.").

130. *See* Lemley et al., *supra* note 39.

on those using the technology,[131] as developers would, the NPE would have to focus on those *not* using the technology.

This is similar to an ex ante patent license that many new patent holders aspire to, but with the added complication that the licensed information is not published for all to see and evaluate. Thus, the parties would have to overcome Arrow's disclosure paradox: the licensing party cannot value the information until it is disclosed, but the information will lose all value if disclosed without a license. Parties typically use non–disclosure agreements to solve this problem. Even so, whether individual inventors and NPEs could get any traction in such a market would remain to be seen. The difficulty they see in the patent market does not bode well: if companies are unwilling to license early when the information is backed by a patent, it is unclear why they would be willing to license when it is not. On the other hand, a pure technology offering might be better taken than a patent license request where infringement must be assumed to support the request. The technology offering would have to be supported by more than just the patented property right.

One might imagine a more nefarious NPE that baits others with trade secrets only to sue them, but such widespread practice is unlikely. For example, a trade secret troll might sue someone for distributing software with unclear reverse engineering restrictions or using onerous non–disclosure agreements.[132] Such behavior would be similar to copyright plaintiffs that sue those who download movies that are offered for download by…those same plaintiffs.[133] Such behavior would be difficult to sustain, however, just as it was for the copyright trolls. Unlike patents, which give the owner the right to exclude infringers regardless of knowledge, companies would be sure to avoid a deceptive licensor once word got out about its bad practices.

E. CHANGING SOFTWARE DELIVERY

The requirement of non–disclosure and anti–reverse engineering provisions would portend a change in how software is delivered. Traditional

131. This is not to say that all NPEs focus only on those who are already infringing. Many attempt to license pre–product technology.

132. David S. Levine & Sharon K. Sandeen, *Here Come the Trade Secret Trolls*, 71 WASH. & LEE L. REV. ONLINE 230, 234 (2014) (describing "the heretofore unknown 'trade secret troll,' an alleged trade secret owning entity that uses broad trade secret law to exact rents via dubious threats of litigation directed at unsuspecting defendants.").

133. *See* Cyrus Farivar, *Prenda Seeded its Own Porn Files via BitTorrent, New Affidavit Argues*, ARS TECHNICA (June 4, 2013), http://arstechnica.com/tech-policy/2013/06/prenda-seeded-its-own-porn-files-via-bittorrent-new-affidavit-shows/.

desktop computer software would likely change little, as such software already includes click–wrap licenses, many of which include anti–reverse engineering clauses. These contracts might change by highlighting the non–disclosure aspects because those terms would be new and unusual.

But other software delivery would change dramatically. Because website operations are revealed software, they would need to be protected by contract—even the ones that do not ordinarily require user accounts or other contracts. Consider, for example, Priceline's reverse auction patent.[134] If Priceline wanted to protect its head start in the market without the patent,[135] it would have had to require all of its users to agree that they would keep the reverse auction secret. But Arrow's information paradox appears again; Priceline might want to advertise how it operates to show potential users the benefits of reverse auctions. But if Priceline showed users how it is different without contractual secrecy, others could copy the format and the site's benefits would not be unique for long.

Software as a service, a hybrid between traditional software and website provisions, would likely become a more prominent form of delivery when using trade secret protections. Software–as–a–service–providers provide software, but they do so on a monthly or annual fee basis. Offline software might look just like "normal" software downloaded for a one–time fee, though it might also be provided only via online forms at a website. Both offline (like Microsoft's Office 360) and online (like online tax preparation services) software as a service already exists. Each comes with a user agreement, either at registration (for online) or at installation/loading (for offline). Those agreements look a lot like traditional software agreements, except they add terms relating to use and management for ongoing payment. What separates software as a service from traditional software, then, is the software provider's ability to remotely block access to the service. At the extremes, such software can track every user's activities, such as use and copying, and even block access to individual features. Tracking who is using the software and how they are using it would add additional information to the mix. This is especially true for managing cross–border compliance.

As a general matter, then, the larger and more widespread the group of users, the more diligence the trade secret owner would need to show. This type of control adds a policing mechanism beyond mere contracting. Leaks can be stopped by blocking the software. Perhaps more important, users can

134. U.S. Patent No. 5,794,207 (filed Sep. 4, 1996).

135. Priceline sued Microsoft to enforce its patent, but settled its case. *Priceline, Microsoft Settle Lawsuit*, CNN MONEY (Jan. 9, 2001), http://money.cnn.com/2001/01/09/technology/priceline/.

be punished for a breach without having to bring suit. The software might even be disabled in certain geographic or company territories. Maintaining control provides additional reasonable precautions beyond a contract, and enhances the ability of the software provider to claim that the features are not generally known.

Here, too, Arrow's paradox is in play. But the solution is the same, as jarring as it may be to current norms: websites and software providers desiring to protect their functionality would have to require agreements before even disclosing how they operate. Their commercials and print advertisements would be reduced to methods that resemble old pharmaceutical ads: "Claritin. I'll ask my doctor!" Further, websites that currently require no user account would likely start requiring such accounts to record contractual agreement before granting any access to important functions. Activity tracking and user name collection may shift important privacy interests.[136]

Websites are not the only software delivered without a traditional click–wrap. Most mobile applications are also delivered without any click–wrap agreement. Those with features that the owner wants to protect might start including an agreement that highlights the non–disclosure aspects.

In one sense, software delivery would not change much. In many cases, the inevitable agreement would simply move to earlier in the process. In another sense, however, the entire ecosystem might change. Less information would be available pre–purchase. Mobile and web contracting might look different, even if the end result is familiar. And contracts would necessarily become even more onerous than they already are, imposing previously unusual restrictions on consumers, including simply turning off the software for perceived violations. This would further mean limited public discussion of features, bugs, and other support items, causing the whole community to suffer.

There is one mixed blessing: the dearth of information means that users will be less able to make a purchasing decision based on ads if those ads would be unlikely to contain secret information. In other words, there would be fewer quick sales based on limited information because there would be *no* information. Instead people may demand evaluation copies and access. These copies would allow them limited access to the software to decide whether it meets the user's needs. Such limited evaluations are often available now, though some require a credit card that gets charged if the user does not cancel by a predetermined time. Evaluations would only

136. The magnitude of such a shift is beyond the scope of this Article; websites are already tracking users regularly.

become more widespread if users had no information on which to decide purchases. Of course, the evaluations would be obtained under confidentiality agreements.

On the one hand, obtaining evaluations is a hassle for users and potentially risky if the provider collects a credit card up front and the user does not cancel. On the other hand, users benefit from being able to evaluate the actual product before purchase, as opposed to relying on overpromising marketing materials.

VI. IMPACTS ON INNOVATION

Trade secret protection does not hinder innovation in the excludability sense. Assuming no patents, any software provider will always be free to design, sell, and compete—even with identical software—so long as that software was not copied or otherwise improperly obtained.

But lack of hindrance seems incomplete. This section considers some potential innovation impacts: a loss of knowledge disclosure, and a potential effect on open source software.

A. A Loss of Learning?

One of the benefits of intellectual property protection is to encourage the dissemination of works so that others can learn from the ideas therein. There are two primary takeaways from this observation, depending on one's views of the intellectual property system generally.

One view, characterized as IP–minimalist, would likely say that changing norms to exert trade secrecy over what would have been publicly available features deprives the world of the learning that could be achieved from those features. This causes costly duplicated effort at best and hinders innovation at worst. People would write programs without trade secrets, and social welfare demands that features remain public so that they can be shared.

The opposing view, characterized as IP–maximalist, would say that intellectual property protection is indeed there to encourage development and dissemination of works, but that appropriable intellectual property protection—namely copyright and patent—have been denied. Thus, the only remotely appropriable option is trade secrecy. Furthermore, without the ability to appropriate value from research and development, there will be far less investment in software. As a result, innovation will be hindered and there will be fewer features for people to learn or discover on their own. Social welfare demands that owners choose the appropriability option that suits them because that will lead to efficient levels of investment. Further,

it was their labor that made the works, and thus the world has no claims to it—especially where the protection is a trade secret that can be protected by contract and not a patent or even a copyright.

The likely effect of this Article's proposal surely lies in the middle ground. As Michael Madison notes in *Open Secrets*,[137] trade secrecy may be a socially beneficial way for groups with similar interests to work together toward improved goals, creating a collection of knowledge known as the commons.[138] While trade secrecy may seem antithetical to the commons, this is not necessarily so. Trade secrecy may be the only way to incentivize the production and, more importantly, the dissemination of software features that would otherwise be held in check. By allowing a group—the software customers—to use the software, a body of knowledge might be sustained.[139] This knowledge might be useful within the group (support and enhancement ideas) or outside the group (sharing of the products created by the software). Thus, even if the software features were protected for a product with a million users, the tradeoff may be acceptable. On the one hand, competitors would lose the ability to easily learn and mimic features. On the other hand, there may be significantly more benefits to wider dissemination of the first product than might be available if, say, the level of reasonable precautions were elevated.

Furthermore, the effect of protecting revealed features on incentives to develop and disseminate innovative products is likely an empirical question that is extremely difficult to answer. This Article is agnostic to the question, because the default law allows for secrecy—but only under certain conditions. The question, then, will be whether the private benefits of secrecy outweigh the costs of those conditions. It likely will not in all cases, which might create a natural experiment to measure innovation effects.

B. HARM TO OPEN SOURCE?

The open source community has long based its view on openness. Open source software is typically created by a person or group that desires to ensure that the software remain freely usable by everyone. Typically, such software is licensed with a proviso that any use shall also include publication of the source code. This is sometimes called "copyleft"—using the power of copyright to force wider dissemination and open sharing.

---

137. Michael Madison, *Open Secrets*, *in* THE LAW AND THEORY OF TRADE SECRECY: A HANDBOOK OF CONTEMPORARY RESEARCH 222 (Rochelle C. Dreyfuss & Katherine J. Strandburg eds., 2011).

138. *Id.* at 225.

139. *See id.* at 227–29.

Presumably, a move to trade secrets would change little. The GNU Public License requires that any software incorporating a GPL licensed project must make the source code available.[140] Software companies wanting to keep their software confidential have thus always been wary of incorporating open source software in their projects. An extension of trade secrecy would not change much.

On the other hand, a primary benefit of open source software is the provision of features that are otherwise available only in non–free software. Expanded trade secrecy would make it more difficult for the open source community to mimic features; this is not surprising given the concerns described above about innovation impacts. Nevertheless, many open source projects drive feature production and would be able to function quite easily without reference to the features of other programs. Even so, open source advocates will likely hate the primary idea of this Article; they disfavor any scheme that overprotects software.

VII. CONCLUSION

This Article posits that with the right type of precautions, revealed software features might be protected by trade secrecy despite the fact that all of the users can see them. The normative considerations of this thought experiment likely depend on one's general view of trade secrets. Skeptics are likely alarmed that such a proposal is even thinkable. Advocates might see little wrong with the concept, even if they wonder whether it would be practically or legally achievable.

This Article explains how such protection might work and explores potential practical and normative effects, it takes no normative views on the competing views. Instead, the goal was to answer a simple question without resolving the final consequences: how might software publishers appropriate their investments in the absence of reliable patent or copyright protection in the visible aspects of software. There are, to be sure, other answers to consider such as first mover, trademarks, and feature based price discrimination.

But trade secrecy is alluring because it has always protected software and it is a known quantity. One need not change the law to achieve the results discussed here, one need only change behavior. This Article has considered how likely (or maybe unlikely) such a scenario might be, and how broad trade secret protection might affect the software licensing

140. Free Software Found., Inc., GNU General Public License § 5(c) (version 3, June 2007), http://www.gnu.org/licenses/gpl-3.0.en.html ("You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy.").

landscape. Trade secret law has long allowed protection of revealed features, but there is little law relating to widely distributed products. To protect such products would likely require a change in the software delivery ecosystem, though many of those changes, such as software as a service, are already underway.