31:3 BERKELEY TECHNOLOGY LAW JOURNAL

2016

Pages

1515

to

1724

Berkeley Technology Law Journal

Volume 31, Number 3

Production: Produced by members of the *Berkeley Technology Law Journal*. All editing and layout done using Microsoft Word.

Printer: Joe Christensen, Inc., Lincoln, Nebraska.
Printed in the U.S.A.
The paper used in this publication meets the minimum requirements of American National Standard for Information Sciences—Permanence of Paper for Library Materials, ANSI Z39.48—1984.

Berkeley Technology Law Journal
University of California
School of Law
3 Boalt Hall
Berkeley, California 94720-7200
btlj@law.berkeley.edu
http://www.btlj.org

BERKELEY TECHNOLOGY LAW JOURNAL

VOLUME 31 NUMBER 3 2016

TABLE OF CONTENTS

ARTICLES

SUBSCRIBER INFORMATION

The *Berkeley Technology Law Journal* (ISSN1086-3818), a continuation of the *High Technology Law Journal* effective Volume 11, is edited by the students of the University of California, Berkeley, School of Law (Boalt Hall) and is published in print three times each year (March, September, December), with a fourth issue published online only (July), by the Regents of the University of California, Berkeley. Periodicals Postage Rate Paid at Berkeley, CA 94704-9998, and at additional mailing offices. POSTMASTER: Send address changes to Journal Publications, University of California, Berkeley Law—Library, LL123 Boalt Hall—South Addition, Berkeley, CA 94720-7210.

Correspondence. Address all correspondence regarding subscriptions, address changes, claims for non-receipt, single copies, advertising, and permission to reprint to Journal Publications, University of California, Berkeley Law—Library, LL123 Boalt Hall—South Addition, Berkeley, CA 94705-7210; (510) 643-6600; JournalPublications@law.berkeley.edu. *Authors*: see section titled Information for Authors.

Subscriptions. Annual subscriptions are $65.00 for individuals and $85.00 for organizations. Single issues are $30.00. Please allow two months for receipt of the first issue. Payment may be made by check, international money order, or credit card (MasterCard/Visa). Domestic claims for non-receipt of issues should be made within 90 days of the month of publication; overseas claims should be made within 180 days. Thereafter, the regular back issue rate ($30.00) will be charged for replacement. Overseas delivery is not guaranteed.

Form. The text and citations in the *Journal* conform generally to the THE CHICAGO MANUAL OF STYLE (16th ed. 2010) and to THE BLUEBOOK: A UNIFORM SYSTEM OF CITATION (Columbia Law Review Ass'n et al. eds., 20th ed. 2015). Please cite this issue of the *Berkeley Technology Law Journal* as 31 BERKELEY TECH. L.J. ___ (2017).

BTLJ ONLINE

The full text and abstracts of many previously published *Berkeley Technology Law Journal* articles can be found at http://www.btlj.org. Our site also contains a cumulative index; general information about the *Journal*; the *Bolt*, a collection of short comments and updates about new developments in law and technology written by BTLJ members; and *BTLJ Commentaries*, an exclusively online publication for pieces that are especially time-sensitive and shorter than typical law review articles.

INFORMATION FOR AUTHORS

The Editorial Board of the *Berkeley Technology Law Journal* invites the submission of unsolicited manuscripts. Submissions may include previously unpublished articles, essays, book reviews, case notes, or comments concerning any aspect of the relationship between technology and the law. If any portion of a manuscript has been previously published, the author should so indicate.

Format. Submissions are accepted in electronic format through the Express*O* online submission system. Authors should include a curriculum vitae and resume when submitting articles, including his or her full name, credentials, degrees earned, academic or professional affiliations, and citations to all previously published legal articles. The Express*O* submission website can be found at http://law.bepress.com/expresso.

Citations. All citations should conform to THE BLUEBOOK: A UNIFORM SYSTEM OF CITATION (Columbia Law Review Ass'n et al. eds., 20th ed. 2015).

Copyrighted Material. If a manuscript contains any copyrighted table, chart, graph, illustration, photograph, or more than eight lines of text, the author must obtain written permission from the copyright holder for use of the material.

DONORS

The *Berkeley Technology Law Journal* and the Berkeley Center for Law & Technology acknowledge the following generous donors to Berkeley Law's Law and Technology Program:

Partners

COOLEY LLP

FENWICK & WEST LLP

ORRICK, HERRINGTON & SUTCLIFFE LLP

Benefactors

COVINGTON & BURLING LLP

SIDLEY AUSTIN LLP

FISH & RICHARDSON P.C.

SKADDEN, ARPS, SLATE, MEAGHER & FLOM LLP & AFFILIATES

KASOWITZ BENSON TORRES & FRIEDMAN LLP

VAN PELT, YI & JAMES LLP

KIRKLAND & ELLIS LLP

WEIL, GOTSHAL & MANGES LLP

LATHAM & WATKINS LLP

WHITE & CASE LLP

MCDERMOTT WILL & EMERY

WILMER CUTLER PICKERING HALE AND DORR LLP

MORRISON & FOERSTER LLP

WILSON SONSINI GOODRICH & ROSATI

WINSTON & STRAWN LLP

Members

BAKER BOTTS LLP

BAKER & MCKENZIE LLP

DURIE TANGRI LLP

GJEL ACCIDENT ATTORNEYS

GTC LAW GROUP LLP & AFFILIATES

GUNDERSON DETTMER STOUGH VILLENEUVE FRANKLIN & HACHIGIAN, LLP

HAYNES AND BOONE, LLP

HOGAN LOVELLS LLP

IRELL & MANELLA LLP

KEKER & VAN NEST LLP

KILPATRICK TOWNSEND & STOCKTON LLP

KNOBBE MARTENS OLSON & BEAR LLP

LEE TRAN & LIANG LLP

MUNGER, TOLLES & OLSON LLP

O'MELVENY & MYERS LLP

PAUL HASTINGS LLP

ROPES & GRAY LLP

SIMPSON THACHER & BARTLETT LLP

TURNER BOYD LLP

WEAVER AUSTIN VILLENEUVE & SAMPSON, LLP

BOARD OF EDITORS 2015–2016

MEMBERSHIP

Vol. 31 No. 3

Associate Editors

WILL BINKLEY
BRITTANY BRUNS
DAPHNE CHEN
YUHAN (ALICE) CHI
KEVIN CHIU
SARA CHUGH
NOA DREYMANN
RAN DUAN
JORDAN FRABONI
JEREMY ISARD
NATASA JANEZ
PAMUDH KARIYAWASAM
HILARY KRASE
JOYCE LI
MATTHEW MCCLELLAN
GAVIN MOLER
CATALINA MONCADA
SHELBY NACINO
JUAN NAZAR
ROBERT OLSEN
EUNICE PARK
JAIDEEP REDDY
ALEJANDRO ROTHAMEL
LUKAS SIMAS
RICHARD SIMS
FERNANDA SOLIS CAMARA
SARAH SUWANDA
JON TANAKA
VALERIE TRUONG
RAOUL GRIFONI-WATERMAN
REID WHITAKER

Members

YASMINE AGELIDIS
NIKI BAWA
KATE BRIDGE
JESSICA BRODSKY
KELSEA CARLSON
STEPHEN CHAO
JOSEPH CHRISTIE
RACHEL CORRIGAN
MICHAEL DEAMER
THOMAS DEC
DARIUS DEHGHAN
ELENA FALLOON
MEGHAN FENZEL
EVAN FERGUSON
WHITNEY FLORIAN
ETHAN FRIEDMAN
ANDREY GAVRILENKO
KAN GU
NOAH GUINEY
BRIAN HALL
ANDREA HALL
JESSICA HOLLIS
JENNIFER HSU
KENSUKE INOUE
HYE JIN KIM
RITHIKA KULATHILA
SARAH KWON
TIFFANY LEUNG
MEI LIU
STASHA LOEZA
SARAH MULLINS
ELI NESS
NATE NGEREBARA
PEGGY NI
JESSICA OGLESBEE
BARCLAY OUDERSLUYS
ROBERT PARIS
MICHELLE PARK
DYLAN PETERSON
KEVIN PORMIR
CHRISTELLE PRIDE
JING XUN QUEK
LEE REDFEARN
MATT RICE
FAITH SHAPIRO
DARINA SHTRAKHMAN
JOSHUA STEELE
EVE TAI
MY THAN
CASEY TONG
ELISSA WALTER
MELISSA WEE
TAMARA WIESEBRON
SHONG YIN
YU ZHAO

BTLJ ADVISORY BOARD

BERKELEY CENTER FOR LAW & TECHNOLOGY 2015–2016

Executive Director

JIM DEMPSEY

Faculty Directors

KENNETH A. BAMBERGER	PETER S. MENELL	PAMELA SAMUELSON
CATHERINE CRUMP	ROBERT P. MERGES	PAUL SCHWARTZ
CHRIS HOOFNAGLE	DEIRDRE MULLIGAN	JENNIFER URBAN
SONIA KATYAL		MOLLY S. VAN HOUWELING

Staff Directors

LOUISE LEE	RICHARD FISK	CLAIRE TRIAS

API COPYRIGHTABILITY BLEAK HOUSE: UNRAVELING AND REPAIRING THE *ORACLE V. GOOGLE* JURISDICTIONAL MESS

Peter S. Menell[†]

ABSTRACT

Like Dickens' tale of Jarndyce and Jarndyce, the *Oracle v. Google* litigation has droned on for what seems like generations in the software industry with no clear end in sight. The litigation is on an especially wasteful and perilous course due to its peculiar jurisdictional posture. As a result of patent infringement allegations lodged in the complaint, the Federal Circuit has exclusive appellate jurisdiction notwithstanding that neither party appealed the rejection of the patent causes of action. Hence, the only issues presented to the Federal Circuit were copyright issues governed by Ninth Circuit—as opposed to Federal Circuit—jurisprudence. The Federal Circuit misinterpreted Ninth Circuit (and general) copyright law, thereby steering the case into a needless fair use retrial.

Congress did not provide a mechanism short of Supreme Court review for ensuring that the Federal Circuit properly interpreted regional circuit law. After tracing the history of the *Oracle v. Google* litigation and critiquing the Federal Circuit's analysis, this Article evaluates a range of potential reforms to the appellate jurisdictional mess presented by software intellectual property litigation and proposes several solutions to this Dickensian predicament.

DOI: https://dx.doi.org/10.15779/Z38B56D426

† Koret Professor of Law and Director of the Berkeley Center for Law & Technology, University of California at Berkeley School of Law. I am grateful to Mark Lemley, Tejas Narechania, David Nimmer, Paul Gugliuzza, and Ryan Vacca for comments on this project, and to Andrea Hall for research assistance.

TABLE OF CONTENTS

I. INTRODUCTION

For more than six years, Oracle and Google have fought a costly and—as of this writing—still unresolved battle over copyright protection for application program interfaces (APIs).[1] The dispute has significant ramifications for much of the software industry,[2] which has been drawn into a high technology version of Jarndyce and Jarndyce. As Charles Dickens explained in the opening chapter of *Bleak House*:

> Jarndyce and Jarndyce drones on. This scarecrow of a suit has, in course of time, become so complicated, that no man alive knows what it means. The parties to it understand it least; but it has been observed that no two Chancery lawyers can talk about it for five minutes, without coming to a total disagreement as to all the premises . . . but Jarndyce and Jarndyce still drags its dreary length before the Court, perennially hopeless.[3]

---

1. An API is:
> a set of subroutine definitions, protocols, and tools for building application software. In general terms, it is a set of clearly defined methods of communication between various software components. It defines methods of communication between various software components and provides access to data of an operating system, application, or other service. A good API makes it easier to develop a computer program by providing all the building blocks, which are then put together by the programmer. An API may be for a web-based system, operating system, database system, computer hardware or software library. An API specification can take many forms, but often includes specifications for routines, data structures, object classes, variables or remote calls.

Application Programming Interface, WIKIPEDIA, https://en.wikipedia.org/wiki/Application_programming_interface (last visited Oct. 8, 2017); Lothar Determann & David Nimmer, *Software Copyright's* Oracle *From the Cloud*, 30 BERKELEY TECH. L.J. 161, 170 (2015) (describing litigation between Google and Oracle over APIs).

2. *See* Nick Wingfield & Quentin Hardy, *Google Prevails as Jury Rebuffs Oracle in Code Copyright Case*, N.Y. TIMES (May 26, 2016), http://www.nytimes.com/2016/05/27/technology/google-oracle-copyright-code.html (quoting representatives of the Electronic Frontier Foundation, Public Knowledge, and a venture capital firm praising the jury's verdict); Michael Hussey, *Copyright Captures APIs: A New Caution For Developers*, TECHCRUNCH (Nov. 3, 2015), https://techcrunch.com/2015/11/03/copyright-captures-apis-a-new-caution-for-developers/ ("Software developers routinely treat APIs as exempt from copyright protection"); Don Clark & Cari Tuna, *Oracle Suit Challenges Google–Silicon Valley Giants Tangle Over Patents, Copyrights Involving Open Programs Android and Java*, WALL ST. J., Aug. 13, 2010, at B1 (noting that the lawsuit was a "surprise move" and "set off shock waves in the Silicon Valley software community").

3. CHARLES DICKENS, BLEAK HOUSE 3 (1853).

Although the *Oracle v. Google* litigation has not yet gone on as long as Jarndyce and Jarndyce in human years, it spans several software generations. Software years are more like dog years.[4] The industry evolves so quickly that companies wither and die if they do not continually innovate. Building on and interoperating with widely adopted software platforms is the lifeblood of Internet age computing and commerce. Yet the *Oracle v. Google* litigation looms, like a dark cloud, over the industry.

The *Oracle v. Google* litigation is on a wasteful and perilous course due to its peculiar jurisdictional posture. In what seems especially ironic in the context of this litigation, Congress established the U.S. Court of Appeals for the Federal Circuit in 1982 for the express purpose of "ending the current legal confusion created by eleven different appellate forums, all generating different interpretations of the patent law."[5] Congress addressed the problem by granting the Federal Circuit exclusive jurisdiction over patent appeals. Yet the Federal Circuit's exclusive appellate jurisdiction over cases involving patent infringement allegations has created a new species of interpretive confusion. In patent cases that contain copyright (or other non–patent) causes of action, regional circuit law binds the Federal Circuit's review of legal questions not exclusively assigned to the Federal Circuit.[6] Moreover, the Federal Circuit will hear the appeals of such non–patent issues even if, as was the circumstance in *Oracle v. Google*, neither party challenged the district court's patent rulings.[7] Congress did not provide a mechanism short of Supreme Court review for ensuring that the Federal Circuit properly interpreted regional circuit law.

---

4. In popular lore, one dog year is the equivalent to seven human years. *See* Erika Mansourian, *How to Calculate Dog Years to Human Years*, AM. KENNEL CLUB (Nov. 16, 2015), http://www.akc.org/content/entertainment/articles/how-to-calculate-dog-years-to-human-years/. The American Veterinary Medical Association offers a more sophisticated formula. The first dog year is equivalent to fifteen human years. The second dog year is equivalent to nine human years. Each additional dog year is equivalent to five human years. *Senior Pets*, AM. VETERINARY MED. ASS'N, https://www.avma.org/public/PetCare/Pages/Senior-Pets.aspx (last visited Oct. 8, 2017).

5. *See* H.R. REP. NO. 96-1307 (1980) (commenting on the legislation that would become the Federal Courts Improvement Act of 1981, Pub. L. 97–164, 96 Stat. 25); *see also* COMM'N ON REVISION OF THE FED. COURT APPELLATE SYS., STRUCTURE AND INTERNAL PROCEDURES: RECOMMENDATIONS FOR CHANGE 15, *as reprinted in* 67 F.R.D. 195, 220 (1975) (discussing the problem of forum shopping in the context of patent cases) [hereinafter HRUSKA COMMISSION REPORT].

6. *See id*. Copyright issues are not exclusively assigned to the Federal Circuit. *See* 28 U.S.C. § 1295 (2012).

7. *See* Atari Games Corp. v. Nintendo of Am., Inc., 897 F.2d 1572, 1575 (Fed. Cir. 1990).

For reasons summarized in this Article and explored in greater depth in a parallel project,[8] the Federal Circuit's 2014 decision in *Oracle v. Google* misinterpreted Ninth Circuit law (and copyright law in general). The unusual jurisdictional posture of the *Oracle v. Google* litigation has produced a Gordian knot of Federal Circuit/Ninth Circuit copyright jurisprudence that cannot easily be untied. Due to the path dependence of the litigation, it is unclear whether the core API copyrightability issue will ever be ripe for Supreme Court review.

Even as *Oracle v. Google* heads back to the Federal Circuit for needless review of a needless second trial, another major software copyright battle governed by this mutant jurisprudence is unfolding in another Northern District of California courtroom.[9] In 2014, Cisco Systems, a leading manufacturer of networking equipment, sued Arista Networks for patent and copyright infringement.[10] As in the *Oracle v. Google* litigation, the copyright claims focus on alleged infringement of Cisco's command line interface (CLI) for configuring, monitoring, and maintaining Cisco devices—an API copyright claim.[11] The district judge, in that case, faces a dilemma—whether to follow the Ninth Circuit's jurisprudence or the Federal Circuit's interpretation of the Ninth Circuit's jurisprudence. In the *Oracle* case, District Judge William Alsup fell into this trap. The result was a reversal of his copyrightability determination. The software industry at large faces a similar dilemma.

This Article examines how software copyright jurisprudence has arrived at this precarious state as well as the larger ramifications for the software industry and appellate intellectual property jurisdiction. Part II summarizes the long and winding history of the *Oracle v. Google* litigation. Part III critiques the Federal Circuit's 2014 copyrightability decision. Part IV traces the possible future pathways for the litigation and explains why the confusing cloud of copyright jurisprudence might continue to loom over the

8. *See* Peter S. Menell, *Rise of the API Copyright Dead?: An Updated Epitaph for Copyright Protection of Network and Functional Features of Computer Software*, 31 HARV. J.L. & TECH. (forthcoming 2018).

9. *See* Quentin Hardy, *In Suit, Cisco Accuses Arista of Copying Work*, N.Y. TIMES: BITS (Dec. 5, 2014), http://bits.blogs.nytimes.com/2014/12/05/in-suit-cisco-accuses-arista-of-copying-work/.

10. *See id.*

11. *See* Cisco Sys. v. Arista Networks, Inc., No. 14-cv-05344-BLF, 2016 U.S. Dist. LEXIS 113285 (N.D. Cal. Aug. 23, 2016); Scott Graham, Cisco v. Arista *IP Battle Starts to Look a Lot Like* Oracle v. Google, RECORDER (Sept. 14, 2017), http://www.therecorder.com/id=1202766017854/Cisco-v-Arista-IP-Battle-Starts-to-Look-a-Lot-Like-Oracle-v-Google.

software industry. Part V explores ways of repairing appellate jurisdiction. Part VI concludes.

II. THE *ORACLE V. GOOGLE* LITIGATION: FROM MICROCOMPUTERS TO THE INTERNET AGE

The *Oracle v. Google* litigation emerged from a dynamic industrial saga that in many ways reflects the evolution of the modern software industry. Section A presents the legislative and jurisprudential backdrop. Section B explores the development of the Java programming language and platform as well as the Android mobile platform. Section C traces the first six years of the *Oracle v. Google* litigation.

A. LEGISLATIVE AND JURISPRUDENTIAL BACKDROP

This Section summarizes the legislative and jurisprudential background to the *Oracle v. Google* litigation.

1. Copyright Legislation

The *Oracle v. Google* saga traces back to Congress's equivocal decision to bring computer software within the scope of copyright protection. Computer software could be expensive to develop and was easily pirated, creating a severe appropriability problem for the nascent, yet critical, commercial software industry.[12] Patent law has long served as the primary intellectual property regime for technological advance.[13] By contrast, copyright law serves as the principal mode of protection for aesthetic creativity.[14] Although computer software—functioning as the gears and levers for digital machines—fell within the technological as opposed to the aesthetic arts, its textual form could more easily be protected through a copyright–type regime, which had long been the primary means of limiting piracy of literary works. Copyright's low threshold for protection, complex scope of protection, broad array of rights, and long duration, however, risked overprotecting software and thereby undermining technological innovation and competition.

12. *See generally* Peter S. Menell, *Tailoring Legal Protection for Computer Software*, 39 STAN. L. REV. 1329 (1987).

13. *See* PETER S. MENELL, MARK A. LEMLEY, & ROBERT P. MERGES, INTELLECTUAL PROPERTY IN THE NEW TECHNOLOGICAL AGE: 2017, VOL I: PERSPECTIVES, TRADE SECRETS, AND PATENTS 168 (2017).

14. *See* PETER S. MENELL, MARK A. LEMLEY, & ROBERT P. MERGES, INTELLECTUAL PROPERTY IN THE NEW TECHNOLOGICAL AGE: 2017, VOL II: COPYRIGHTS, TRADEMARKS & STATE IP PROTECTIONS 498–500 (2017).

The software protection controversy emerged at an inopportune time. Congress had been working for nearly two decades to overhaul the Copyright Act of 1909 and was nearing closure in the early to mid–1970s.[15] Faced with the difficult challenge of fitting computer and other new information technologies under the existing umbrella of intellectual property protection, Congress established the National Commission on New Technological Uses of Copyrighted Works (CONTU) to study the implications of computer software and recommend revisions to federal intellectual property law.[16]

As a stopgap, Congress included software within the scope of copyright protection in the Copyright Act of 1976 ("1976 Act"),[17] but subject to foundational limitations set forth in § 102(b): "In no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied in such work."[18] The legislative history noted that:

> [s]ome concern has been expressed lest copyright in computer programs should extend protection to the methodology or processes adopted by the programmer, rather than merely to the "writing" expressing his ideas. Section 102(b) is intended, among other things, to make clear that the expression adopted by the programmer is the copyrightable element in a computer program, and that the actual processes or methods embodied in the program are not within the scope of the copyright law.[19]

After conducting extensive hearings and receiving expert reports, a majority of CONTU's blue–ribbon panel of copyright authorities and interest group representatives concluded that the intellectual work

---

15. *See* Peter S. Menell, *In Search of Copyright's Lost Ark: Interpreting the Right to Distribute in the Internet Age*, 59 J. COPYRIGHT SOC'Y U.S.A. 1, 31–32 (2011).

16. Act of Dec. 31, 1974, Pub. L. No. 93-573, § 201, 88 Stat. 1873.

17. The Act includes "literary works" within the class of "works of authorship." *See* 17 U.S.C. § 102(a)(1) (2012). The House Report explains that:

> [t]he term "literary works" does not connote any criterion of literary merit or qualitative value: it includes catalogs, directories, and similar factual, reference, or instructional works and *compilations of data*. It also includes *computer data bases*, and *computer programs* to the extent that they incorporate authorship in the programmer's expression of original ideas, as distinguished from the ideas themselves.

H.R. REP. NO. 94-1476, at 53–54 (1976) (emphasis added).

18. 17 U.S.C. § 102(b) (2012).

19. *See* H.R. REP. NO. 94-1476, at 57 (1976).

embodied in computer software should be protected under copyright law, notwithstanding the fundamental principle that copyright cannot protect "any idea, procedure, process, system, method of operation, concept, principle, or discovery"[20] and the Supreme Court's foundational decision in *Baker v. Selden*.[21] CONTU recommended two modest changes to the 1976 Act: (1) defining a computer program as "a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result"; and (2) allowing "the rightful possessor of a copy of a computer program" to run the program and to make a backup copy of the program without infringement liability.[22] Congress implemented CONTU's recommendation in its 1980 amendments to federal copyright law with one confusing wording change.[23]

The CONTU Final Report explained that while "one is always free to make a machine perform any conceivable process (in the absence of a patent) . . . one is not free to take another's program," subject to copyright's limiting doctrines—originality and the idea–expression dichotomy.[24] The Report further explained that:

> The "idea-expression identity" exception provides that copyrighted language may be copied without infringing when there is but a limited number of ways to express a given idea. This rule is the logical extension of the fundamental principle that copyright cannot protect ideas. In the computer context this means that when specific instructions, even though previously copyrighted, are the only and essential means of accomplishing a

20. 17 U.S.C. § 102(b) (2012).

21. *See* NAT'L COMM'N ON NEW TECHNOLOGICAL USES OF COPYRIGHTED WORKS, FINAL REPORT 1 (1979), http://digital-law-online.info/CONTU/PDF/index.html [hereinafter CONTU FINAL REPORT]. *But see id*. at 27–37 (Commissioner Hersey, dissenting) (arguing that "forcible wrenching" would be required to protect computer programs under the copyright law); *id*. at 37–38 (Commissioner Karpatkin, dissenting) (same); *cf. id*. at 26–27 (Commissioner Melville B. Nimmer, concurring) (warning that CONTU recommendations might take copyright law "beyond the breaking point," converting it into a general misappropriation law).

22. *See id*. at 12.

23. *See* Act of Dec. 12, 1980, Pub. L. No. 96-517, 94 Stat. 3007, 3028 (codified at 17 U.S.C. §§ 101, 117 (2012)). For reasons that were not explained in the legislative history of the 1980 amendments, Congress narrowed CONTU's category of "rightful possessor" to "rightful owner." *See* 2 MELVILLE B. NIMMER & DAVID NIMMER, NIMMER ON COPYRIGHT § 8.08[B][1][c][ii] (2017).

24. *See* CONTU FINAL REPORT, *supra* note 21, at 20. Courts have treated the CONTU FINAL REPORT as legislative history for the 1980 amendments to the 1976 Act. *See* Vault Corp. v. Quaid Software Ltd., 847 F.2d 255, 260–61 (5th Cir. 1988); Apple Comput., Inc. v. Franklin Comput. Corp., 714 F.2d 1240, 1252 (3d Cir. 1983).

> given task, their later use by another will not amount to an infringement.[25]

Thus, while recognizing important limitations on copyright protection for computer software, including the § 102(b) limitations, Congress intended that software programmers would garner protection for their programming design and coding choices to the extent that the expression was separable from the underlying ideas. In this way, the general programming ideas and unoriginal programming choices remain free for others to use while the creative effort in particularized programming choices and compilations, especially in complex programs, gains protection from copyists.

Interpreting and applying the idea–expression dichotomy in software cases, like other important "common law" copyright doctrines,[26] fell to the courts.

2. Copyright Jurisprudence

The rapid growth of the microcomputer and consumer software industries fueled more than a decade of litigation centered on the scope of copyright protection for computer software. These cases spanned Apple's litigation to bar clones of its breakthrough Apple II computer, Apple's effort to block Microsoft Windows from competing with the Macintosh's graphical user interface, mobile phone companies' copyright claims to codes for cellular phone networks, Sega's effort to control access to its Genesis videogame console, and Lotus's effort to control the menu command hierarchy of the Lotus 1–2–3 spreadsheet program. These cases, and many other software copyright battles, centered on the idea–expression dichotomy: to what extent could platform innovators protect application program interfaces through copyright protection?

The early cases suggested a broad scope of copyright protection for computer software and interoperable features. The first major software copyright cases pitted Apple Computer Corporation, then a young, break–out microcomputer company, against brash competitors offering inexpensive "interoperable" Apple II clones.[27] The clone makers quickly entered the market by copying, bit by bit, Apple's operating system and

25. *Id*. at 20 (footnote omitted).

26. *See generally* Peter S. Menell, *The Mixed Heritage of Federal Intellectual Property Law and Ramifications for Statutory Interpretation*, *in* INTELLECTUAL PROPERTY AND THE COMMON LAW 63 (Shyamkrishna Balganesh ed., 2013).

27. *See* Apple Comput., Inc. v. Franklin Comput. Corp., 545 F. Supp. 812 (E.D. Pa. 1982), *rev'd*, 714 F.2d 1240 (3d Cir. 1983); Apple Comput., Inc. v. Formula Int'l, Inc., 562 F. Supp. 775 (C.D. Cal. 1983), *aff'd*, 725 F.2d 521 (9th Cir. 1984).

application programs. The defendants argued that copyright protection did not extend to non–human readable (object code) formats of computer software and that the idea–expression doctrine barred copyright protection for operating system programs. They further argued that copyright protection should not stand in the way of selling computers that can run programs written for the Apple II.

The courts had little difficulty finding that copying the entirety of sophisticated computer programs constituted copyright infringement. In reaching these findings, however, the courts went overboard in their dicta. Addressing the defendant's interoperability argument, the Third Circuit opined that "total compatibility with independently developed application programs . . . is a commercial and competitive objective which does not enter into the somewhat metaphysical issue of whether particular ideas and expressions have merged."[28] Since two entirely different programs can achieve the same "certain result[s]"—for example, generate the same set of protocols needed for interoperability—the court was not justified in making such an expansive statement about the scope of copyright protection for computer program elements.[29] Given the verbatim copying of millions of bits of object code, there was no need to address the interoperability issue. The defendant offered no explanation of which elements of the program were protectable and which were not.

The next major software copyright appellate decision also arose from the Third Circuit. In *Whelan Associates, Inc. v. Jaslow Dental Laboratory, Inc.*,[30] a computer programmer sued the dental laboratory, for which it had developed a computer program for managing its bookkeeping functions, for copyright infringement after an officer of the laboratory set out to create a version of the program that would run on other computer systems. The competing software did not literally copy Whelan's code, but there were overall structural similarities between the two programs. To distinguish protectable expression from unprotectable idea, the court reasoned:

> *[T]he purpose or function of a utilitarian work would be the work's idea, and everything that is not necessary to that purpose or function would be part of the expression of the idea.* Where

28. *See* Apple Comput., Inc. v. Franklin Comput. Corp., 714 F.2d 1240, 1253 (3d Cir. 1983).

29. CONTU was clear that "[o]ne is always free to make the machine do the same thing as it would if it had the copyrighted work placed in it, but only by one's own creative effort rather than by piracy." *See* CONTU FINAL REPORT, *supra* note 21, at 21.

30. 797 F.2d 1222 (3d Cir. 1986).

> there are many means of achieving the desired purpose, then the particular means chosen is not necessary to the purpose; hence, there is expression, not idea.[31]

In applying this rule, the court defined the idea as "the efficient management of a dental laboratory," which could be expressed in countless ways.[32] Drawing the idea–expression dichotomy at such a high level of abstraction implies an expansive scope of copyright protection. Furthermore, the court's conflation of merger analysis and the idea–expression dichotomy implicitly allows copyright protection of procedures, processes, systems, and methods of operation that § 102(b) expressly excludes.

Although the case did not directly address copyright protection for interoperability protocols, the court's mode of analysis dramatically expanded the scope of copyright protection for computer programs. If everything below the general purpose of the program was protectable under copyright, then it would follow that particular protocols were protectable because there would be alternative means to accomplish the program's general purpose. Such a result would effectively bar competitors from developing interoperable programs and computer systems.

Commentators roundly criticized the *Whelan* test,[33] and other courts began to refine the scope of copyright protection to comport with the fundamental principles (including limitations) of copyright protection. A few months after the *Whelan* decision, the Fifth Circuit confronted a similar claim of copyright infringement based upon structural similarities between two programs designed to provide cotton growers with information regarding cotton prices and availability, accounting services, and a means for conducting cotton transactions electronically.[34] In declining to follow the *Whelan* approach, the court found that the similarities in the programs

---

31. *Id.* at 1236 (emphasis in original) (citations omitted).

32. *Id.*

33. *See, e.g.*, Donald S. Chisum et al., *LaST Frontier Conference Report on Copyright Protection of Computer Software*, 30 JURIMETRICS J. 15, 20–21 (1989); Steven R. Englund, *Idea, Process, or Protected Expression?: Determining the Scope of Copyright Protection of the Structure of Computer Programs*, 88 MICH. L. REV. 866, 881 (1990); Peter S. Menell, *Analysis of the Scope of Copyright Protection for Application Programs*, 41 STAN. L. REV. 1045 (1989); David Nimmer, Richard L. Bernacchi & Gary N. Frischling, *A Structured Approach to Analyzing the Substantial Similarity of Computer Software in Copyright Infringement Cases*, 20 ARIZ. ST. L.J. 625, 629–34 (1988).

34. Plains Cotton Coop. Ass'n v. Goodpasture Comput. Serv., Inc., 807 F.2d 1256 (5th Cir. 1987).

were dictated largely by standard practices and forms in the cotton market—what the court called "externalities"—which constitute unprotectable ideas.[35]

In 1992, the Second Circuit adapted Learned Hand's seminal abstraction–filtration–comparison framework[36] to computer software analysis.[37] Computer Associates (CA), a leading mainframe software provider, had developed SCHEDULER, a job–scheduling program that worked with three IBM mainframe computers. Part of the success of this program was that it had a subcomponent, called ADAPTER, which would interoperate with any of the three IBM mainframes (DOS/VSE, MVS, and VM/CMS). As a result, the user did not need to customize its programs for each of the IBM mainframes. ADAPTER ensured that programs written for SCHEDULER would interoperate with any of the three IBM mainframes.

In developing a competing job scheduling computer program (ZEKE), which had its own code layer (OSCAR) for interoperating with the three IBM mainframes, Altai relied on James Arney, a former CA programmer. Unbeknownst to Altai's management, Arney improperly copied 30% of OSCAR from CA's ADAPTER program.[38] When Altai's executives learned of the illicit copying, the company initiated a clean–room[39] rewrite of the program. Drawing on the *Whelan* decision, CA challenged the revised version of ZEKE based on structural similarities. The district court criticized *Whelan*'s "simplistic test" for determining similarity between computer programs,[40] rejecting the notion that there is but one idea per program and that as long as there were alternative ways of expressing that one idea, then any particular version was protectable under copyright law.

35. *Id.* at 1262 (finding the commonly used "cotton recap sheet," for summarizing basic transaction information, to be unprotectable). The court was persuaded by the decision in *Synercom Technology, Inc. v. University Computing Co.*, 462 F. Supp. 1003, 1012–13 (N.D. Tex. 1978), which analogized the "input formats" of a computer program (the organization and configuration of information to be inputted into a computer) to the "figure-H" pattern of an automobile stick shift.

36. *See* Nichols v. Universal Pictures Corp., 45 F.2d 119 (2d Cir. 1930).

37. *See* Comput. Assocs. Int'l v. Altai, Inc., 982 F.2d 693 (2d Cir. 1992).

38. *Id.* at 699. Altai accepted responsibility for copyright infringement based on Arney's misdeeds and was ordered to pay $364,444 in damages. *See id.* at 696.

39. A clean room process insulates programmers from copyright protected code in producing code that accomplishes the same functions as a target program based solely on the functional specifications. Such a process ensures that a program is independently written and hence not copied except with regard to unprotectable elements. *See generally* P. Anthony Sammi, Christopher A. Lisy & Andrew Gish, *Good Clean Fun: Using Clean Room Procedures in Intellectual Property Litigation*, 25 INTELL. PROP. & TECH. L.J. 3 (2013).

40. 775 F. Supp. 544, 558 (E.D.N.Y. 1991).

Focusing on the various levels of the computer programs at issue, the court determined that the similarities between the programs were dictated by external factors—such as the interface specifications of the IBM operating system and the demands of functionality—and hence no protected code was infringed.

The Second Circuit decision fleshed out the analytical framework for determining copyright infringement of computer program code:

> In ascertaining substantial similarity . . . a court would first break down the allegedly infringed program into its constituent structural parts. Then, by examining each of these parts for such things as incorporated ideas, expression that is necessarily incidental to those ideas, and elements that are taken from the public domain, a court would then be able to sift out all non-protectable material. Left with a kernel, or perhaps kernels, of creative expression after following this process of elimination, the court's last step would be to compare this material with the structure of an allegedly infringing program.[41]

The court's abstraction–filtration–comparison test recognized that an idea could exist at multiple levels of a computer program and not solely at the most abstract level. Furthermore, the ultimate comparison is not between the programs as a whole but rather between a program's protectable elements and those that allegedly copy them. Of most importance with regard to fostering interoperability, the court held copyright protection did not extend to those program elements where the programmer's freedom to choose is:

> circumscribed by extrinsic considerations such as (1) mechanical specifications of the computer on that a particular program is intended to run; (2) compatibility requirements of other programs with which a program is designed to operate in conjunction; (3) computer manufacturers' design standards; (4) demands of the industry being serviced; and (5) widely accepted programming practices within the computer industry.[42]

---

41. *Altai*, 982 F.2d at 706.
42. *Id.* at 709–10. The court observed that:

    > [w]hile, hypothetically, there might be a myriad of ways in which a programmer may effectuate certain functions within a program–*i.e.*, express the idea embodied in a given subroutine–efficiency concerns may so narrow the practical range of choice as to make only one or two forms of expression workable operations.

Id. at 708.

Directly rejecting the dictum in *Apple v. Franklin*,[43] the Second Circuit recognized that external factors such as interface specifications, de facto industry standards, and accepted programming practices are not protectable under copyright law. The Second Circuit test evaluates these external factors at the time of the allegedly infringing activities (i.e., ex post), not at the time that the first program is written.[44]

Commentators warmly embraced the *Altai* decision,[45] and courts have universally adopted the abstraction–filtration–comparison.[46] The Ninth Circuit's decision in *Sega Enterprises Ltd. v. Accolade*[47] expressly recognized the legitimacy of deciphering and copying lockout codes for purposes of developing interoperable products. Sega developed a successful video game platform (Genesis) for which it licensed access to video game developers. Accolade, a manufacturer of video games, wanted to distribute versions of its games on the Genesis platform. It did not, however, want to limit distribution exclusively to Genesis, as Sega required. Rather than license Sega's code, Accolade reverse engineered the access code through a painstaking effort that entailed making hundreds of intermediate copies of Sega's computer code. Accolade then incorporated only the code (approximately 25 bytes in games containing between 500,000 and 1.5 million bytes) necessary to achieve interoperability with the Genesis platform.[48]

Sega sued Accolade for copyright infringement. Given the relatively small amount of Sega code in the Accolade game cartridges, Sega focused its copyright claim on the making of intermediate copies of its full computer program made during the process of reverse engineering. The district court

43. *See* Apple Comput., Inc. v. Franklin Comput. Corp., 714 F.2d 1240, 1251 (3d Cir. 1983).

44. The court emphasized that the first to write a program for a particular application should not be able to "'lock up' basic programming techniques as implemented in programs to perform particular tasks." *Altai*, 982 F.2d at 712 (quoting Menell, *supra* note 33).

45. *See* David Bender, Computer Associates v. Altai*: Rationality Prevails*, 9 COMPUTER LAW. 1 (1992); Peter S. Menell, *The Challenges of Reforming Intellectual Property Protection for Computer Software*, 94 COLUM. L. REV. 2644, 2652 (1994).

46. *See* Peter S. Menell, *Envisioning Copyright Law's Digital Future*, 46 N.Y.L. SCH. L. REV. 63, 84–85 (2002); Mark A. Lemley, *Convergence in the Law of Software Copyright?*, 10 BERKELEY TECH. L.J. 1 (1995).

47. 977 F.2d 1510 (9th Cir. 1992) [hereinafter *Sega Enters. II*].

48. *Id.* at 1516.

rejected Accolade's argument that such intermediate copies constituted fair use and granted a preliminary injunction.[49]

The Ninth Circuit reversed, holding that "the functional requirements for compatibility with the Genesis [video game console are] aspects of Sega's programs that are not protected by copyright."[50] Building on that foundation, the court ruled that "disassembly of object code in order to gain an understanding of the ideas and functional concepts embodied in the code is a fair use that is privileged by section 107 of the Act."[51] The court determined that policies underlying the Copyright Act authorize disassembly of copyrighted object code and the making of intermediate copies to discover unprotectable elements of code.[52] The Ninth Circuit reaffirmed and expanded this doctrine in *Sony Computer Entertainment, Inc. v. Connectix Corp*.[53]

The Northern District of California and the Ninth Circuit applied the *Altai* framework to the graphical user interface features of a computer program in *Apple Computer, Inc. v. Microsoft Corp*.[54] Apple Computer alleged that Microsoft's Windows operating system infringed Apple's copyrights in the desktop graphical user interface of its Macintosh computer system. A licensing agreement authorizing the defendants' use of aspects of Apple's graphical user interface muddied the copyright issue.[55] The court determined, however, that the licensing agreement was not a complete defense to the copyright infringement claims and consequently analyzed the scope of copyright protection for a range of audiovisual display elements.[56]

The district court found that the unlicensed similarities between Apple's works and Microsoft's Windows were either unprotectable or subject to at least one of copyright law's limiting doctrines. In evaluating the compilation of these elements as a whole, the court applied the "virtual

49. *See* Sega Enters. Ltd. v. Accolade, Inc., 785 F. Supp. 1392, 1397–1400 (N.D. Cal. 1992), *rev'd*, 977 F.2d 1510 (9th Cir. 1992).

50. *Sega Enters. II*, 977 F.2d at 1522 (citing 17 U.S.C. § 102(b) (2012)).

51. *Id.* at 1517–18.

52. *See id*.

53. 203 F.3d 596 (9th Cir. 2000).

54. 799 F. Supp. 1006 (N.D. Cal. 1992), *aff'd in part, rev'd in part*, 35 F.3d 1435 (9th Cir. 1994).

55. *See id*. at 1015, 1031–32, 1041.

56. *See* Apple Comput., Inc. v. Microsoft Corp., 759 F. Supp. 1444 (N.D. Cal. 1991); Apple Comput., Inc. v. Microsoft Corp., 717 F. Supp. 1428 (N.D. Cal. 1989); Apple Comput., Inc. v. Microsoft Corp., 709 F. Supp. 925, 930 (N.D. Cal. 1989).

identity" standard[57] and determined that no infringement had occurred. On appeal, the Ninth Circuit affirmed the district court's dissection of Apple's graphical user interface to determine which elements are protectable, filtering of unprotectable elements, and application of the "virtual identity" standard.[58]

The copyrightability of command systems for computer software arose in litigation over spreadsheet technology. Building upon the success of the VisiCalc program developed for the Apple II computer, Lotus Corporation marketed a spreadsheet program for the IBM PC platform—Lotus 1–2–3—which offered many of VisiCalc's features and commands while integrating charting and database capabilities.[59] Lotus 1–2–3 quickly became the market leader for spreadsheets running on IBM and IBM–compatible machines.[60] As a result, knowledge of the program became especially valuable for accountants and managers. The 1–2–3 command hierarchy provided a logically structured menu of more than 200 commands and enabled users to develop customized programs (called macros) for their particular accounting and business planning functions. These investments locked users into the 1–2–3 command structure as their library of macros grew.[61] By the late 1980s, software developers seeking to enter the spreadsheet market could not ignore the large premiums that many consumers placed on transferring their investments in the 1–2–3 system to a new spreadsheet environment, even where a new spreadsheet product offered significant technical improvements over the Lotus spreadsheet.[62]

After three years of intensive development efforts, Borland International, developer of several successful software products including

57. The Ninth Circuit developed the heightened "virtual identity" standard for evaluating thinly protected works such as compilations of simple, narrowly protected elements, including the visual layout of a day planner (comprising a calendar and ruled lines), *see* Harper House, Inc. v. Thomas Nelson, Inc., 889 F.2d 197 (9th Cir. 1989), and the audiovisual elements of a karate videogame, *see* Data E. USA, Inc. v. Epyx, Inc., 862 F.2d 204 (9th Cir. 1988).

58. Apple Comput, Inc. v. Microsoft Corp., 35 F.3d 1435 (9th Cir. 1994).

59. *See* D. J. Power, *A Brief History of Spreadsheets*, DSSRESOURCES.COM (Aug. 30, 2004), http://dssresources.com/history/sshistory.html; Pamela Samuelson, *Computer Programs, User Interfaces, and Section 102(b) of the Copyright Act of 1976: A Critique of* Lotus v. Paperback, 6 BERKELEY TECH. L.J. 209, 245 n.145 (1991).

60. *See id.*

61. *See* Neil Gandal, *Hedonic Price Indexes for Spreadsheets and an Empirical Test for Network Externalities*, 25 RAND J. ECON. 160 (1994).

62. *See* Mike Hogan, *Product Outlook: Fresh from the Spreadsheet Oven*, PCWORLD, Feb. 1988, at 100, 102; Lawrence J. Magid, *'Surpass' Spreadsheet Program Lives Up to Name, Beats Lotus 1-2-3*, WASH. POST, Apr. 25, 1988, at 26.

Turbo Pascal and Sidekick, introduced Quattro Pro, its entry into the spreadsheet market. Quattro Pro made substantial design and operational improvements and earned accolades in the computer product review magazines.[63] Quattro Pro offered a new interface for its users, which many purchasers of spreadsheets preferred over the 1–2–3 interface. Nonetheless, because of the large number of users already familiar with the 1–2–3 command structure and those who had made substantial investments in developing macros to run on the 1–2–3 platform, Borland considered it essential to offer an operational mode based on the 1–2–3 command structure as well as macro compatibility. Borland's visual representation of the 1–2–3 command mode substantially differed from the 1–2–3 screen displays.

The lower court held that a menu command structure was protectable if there were many such structures available.[64] The court also found that Borland was not permitted to achieve macro compatibility with the 1–2–3 product, distinguishing the treatment of external constraints noted in the *Altai* decision on the ground that such constraints had to exist at the time that the first program was created.[65] The First Circuit reversed, holding that the menu command hierarchy was a "method of operation" that fell within the copyright exclusion set forth in § 102(b).[66] The U.S. Supreme Court

63. *See Spreadsheet; Borland International Inc.'s Quattro Pro for Windows and Quattro Pro 4.0 for DOS*, PC-COMPUTING, Dec. 1992, at 140 ("No doubt about it: Quattro Pro for DOS is the best DOS spreadsheet there is. Period."); *Borland's Quattro Pro Tops 2.5 Million Units Shipped*, BUS. WIRE, July 1, 1992 ("Since its introduction in October 1989, Quattro Pro has won an unprecedented 42 industry awards and honors worldwide from users and product reviewers."); Lewis Peter, *Software Review, Quattro Pro 4.0; Borland International Inc.'s Spreadsheet Software*, COMPUTER SHOPPER, June 1992, at 536 ("Quattro Pro 4.0 simply shames other DOS-based spreadsheets, especially Lotus 1-2-3 r2.").

64. *See* Lotus Dev. Corp. v. Borland Int'l, Inc., 831 F. Supp. 202, 215 (D. Mass. 1993) ("[A]lthough functional considerations may have some effect on the design of a menu tree, they do not impose any practical limitation on the possible forms of expression to a number far enough short of infinity that any way of expressing the number in English words has come into common usage"), *rev'd*, 49 F.3d 807 (1st Cir. 1995), *aff'd without opinion by equally divided court*, 516 U.S. 233 (1996).

65. *See* Lotus Dev. Corp. v. Borland Int'l, Inc., 799 F. Supp. 203, 211–14 (D. Mass. 1992), *rev'd*, 49 F.3d 807 (1st Cir. 1995), *aff'd without opinion by equally divided court*, 516 U.S. 233 (1996).

66. Lotus Dev. Corp. v. Borland Int'l., Inc., 49 F.3d 807, 814–15 (1st Cir. 1995), *aff'd without opinion by equally divided court*, 516 U.S. 233 (1996).

granted certiorari and affirmed without opinion by an equally divided vote.[67]

Subsequent appellate decisions reached similar outcomes, although they have not fully subscribed to the First Circuit's reasoning.[68] Thus, after an inauspicious start, the federal courts implemented a balanced framework for both protecting computer software against piracy and interpreting the idea–expression dichotomy in such a way to ensure that copyright law does not extend to functional features of computer technology. Following resolution of the first API copyright war, the software engineering community believed that copyright law did not protect high–level functions, labeling conventions, and APIs.[69] Software copyright litigation subsided, and there were no new major API copyright judicial decisions until *Oracle v. Google* more than a decade later.

B. ROOTS OF THE *ORACLE V. GOOGLE* LITIGATION

The Java platform emerged with great fanfare in the mid–1990s. Just as the software copyright battles subsided, the open source movement was gaining traction, and the Internet was opening for business. The rise of the open source movement and the emergence of the Internet brought about more open and collaborative software development strategies. Sun Microsystems, the developer of Java, embraced the open bandwagon, which

67. Lotus Dev. Corp. v. Borland Int'l, Inc., 516 U.S. 233 (1996) (Justice Stevens recused himself from participation in consideration of the case). As a result, the First Circuit's *Lotus* decision remained the law in the First Circuit but did not bind other circuits.

68. *See* Mitel, Inc. v. Iqtel, Inc., 124 F.3d 1366, 1373–74 (10th Cir. 1997) (holding that a computer system for automating the selection of telephone long–distance carrier and remotely activating optional telecommunications features lacked the minimal degree of creativity to qualify for copyright protection and should be denied copyright protection under the *scènes à faire* doctrine because such systems are largely dictated by external factors including compatibility requirements and industry practices; but declining to hold that menu command hierarchies are categorically excluded from copyright protection); MiTek Holdings, Inc. v. ARCE Eng'g Co., 89 F.3d 1548, 1556–57 (11th Cir. 1996) (holding that the menu and submenu command structure of a software program for designing wood trusses for the framing of building roofs was uncopyrightable under § 102(b) of the Copyright Act because it represents a process).

69. *See* Brian Proffitt, *The Impact of Oracle's Defense of API Copyrights*, ITWORLD (Aug. 23, 2011), http://www.itworld.com/article/2738675/mobile/the-impact-of-oracle-s-defense-of-api-copyrights.html ("Historically, APIs have been regarded as not falling under copyright–the reasoning being that APIs are not creative implementations but rather statements of fact"); Michael Hussey, *Copyright Captures APIs: A New Caution For Developers*, TECHCRUNCH (Nov. 3, 2015), https://techcrunch.com/2015/11/03/copyright-captures-apis-a-new-caution-for-developers/ ("Software developers routinely treat APIs as exempt from copyright protection.").

fueled widespread adoption of Java for website development. A decade after Java's release, Google foresaw the need to develop a robust open mobile operating system. It drew heavily upon Java's widely adopted language and API packages in developing Android.

1. The Java Platform

The phenomenal success of the Java programming language and platform was to a large extent serendipitous. To understand Java's development, it is necessary to understand the origins of Sun Microsystems in the 1980s. Sun quickly earned a reputation for its high–end computer workstations and its quirky, innovative culture.[70] Its technology fueled Silicon Valley's meteoric rise. Sun went public in 1986 under the stock symbol SUNW for Sun Workstations (later Sun Worldwide),[71] and hit $1 billion in revenues in 1988, a record for a Silicon Valley company.[72] Thanks to its reputation for cutting–edge products and engineer–friendly culture, the company attracted a talented and eclectic group of engineers and programmers.

Sun's revenues and market value grew steadily from its founding into the mid–1990s and skyrocketed during the dot–com boom that followed.[73] Flush with venture capital investment, many start–ups wanted the best workstations and servers for their engineering and programming teams.

Sun's foray into developing a new programming language began in 1990 as a skunkworks project triggered by an effort to retain top programmers. It initially aimed at developing a new generation of software to replace Sun's C and C++ APIs and tools.[74] The project evolved into developing a computer language and handheld device to control digital

70. *See* David Bank, *The Java Saga*, WIRED (Dec. 1, 1995), http://www.wired.com/1995/12/java-saga/ (noting that while "Sun's machines had a reputation for being too complicated, too ugly, and too nerdy for mass consumption," its leadership was willing "to loosen[] the reins on some of its most precocious [programmer] talent.").

71. *See* John Letzing, *In 1986, Sun Led the Way for Future Tech Giants*, MARKETWATCH (Oct. 19, 2009, 10:00 AM), http://www.marketwatch.com/story/sun-was-first-to-go-public-is-first-to-disappear-2009-10-19.

72. *See* Laura Lambert, *William Joy (1954–), Programmer; Founder of Sun Microsystems*, *in* THE INTERNET: BIOGRAPHIES 138 (Hilary W. Poole ed., 2005).

73. *See* Letzing, *supra* note 71; Lee Devlin, *The Sun Also Sets*, K0LEE.COM (Oct. 2, 2009), http://k0lee.com/2009/10/sun-also-sets/ (tracing Sun's meteoric stock rise from 1982 to 2000 and subsequent fall).

74. *See* Bank, *supra* note 70; Nenad Dumanovic, *After 20 Years, the Java Phenomenon Just Keeps Going*, CERTIFICATION MAG. (Aug. 19, 2015), http://certmag.com/20-years-java-phenomenon-just-keeps-going/.

consumer products (such as televisions) and computers.[75] Such a language needed to be scaled for embedded systems—computer systems with a dedicated function within other systems.[76]

The team focused on developing a distributed computing environment for set–top boxes, interactive TV, and videocassette recorders through a wireless network.[77] Such a system would require a more compact footprint and hence would have more limited functionality than general purpose computing systems. By 1993, the software (codenamed Oak) was integrated into a versatile, handheld interactive TV device. Sun was unable, however, to interest consumer electronics or cable companies.[78]

Just when the project looked doomed, Bill Joy, one of Sun's founders, saw the opportunity to adapt Oak for the nascent, but promising, World Wide Web.[79] Joy realized that Oak could be repurposed to program Web pages as opposed to consumer devices. "Java," the renamed project, aimed to develop a simple, lean, platform–independent, real–time, embeddable, multitasking programming language for Web functionality. Java had a similar syntax to the widely used C language, but was far more compact, efficient, and secure. Of perhaps greatest importance, Java enabled "write once, run anywhere" (WORA) functionality—Java applets could run on Apple, Windows, or UNIX machines. Java also enabled real–time interactivity, multimedia, and animation, which greatly enhanced the dynamism of Web pages, enabling users to interact with websites in new and exciting ways.

With the experimental new software platform reaching fruition, Sun faced a conundrum. Although Sun had promoted open standards for software interfaces,[80] this project would require free release of a software

75. *See History of the Java Programming™ Language*, WIKIBOOKS, https://en.wikibooks.org/wiki/Java_Programming/History (last visited Oct. 8, 2017).

76. *See* Pamela Samuelson, *Foreword*, 13 BERKELEY TECH. L.J. 809, 816 (1998); Laura McNeill Hutcheson, *The Exclusion of Embedded Software and Merely Incidental Information from the Scope of Article 2B: Proposals for New Language Based on Policy and Interpretation*, 13 BERKELEY TECH. L.J. 977, 983–94 (1998); Embedded system, WIKIPEDIA, https://en.wikipedia.org/wiki/Embedded_system (last visited Oct. 8, 2017).

77. *See* Laura Lambert, *James Gosling (1956–), Inventor of Java*, *in* THE INTERNET: BIOGRAPHIES 132–36 (Hilary W. Poole ed., 2005).

78. Bank, *supra* note 70; Lambert, *supra* note 77, at 133–34.

79. *See* Lambert, *supra* note 72, at 138.

80. Sun Microsystems has been the leading member of the American Committee for Interoperable Systems (ACIS), an early lobbying organization advocating open platforms. *See* JONATHAN BAND & MASANOBU KATOH, INTERFACES ON TRIAL: INTELLECTUAL

implementation—i.e., the full program. Marc Andreessen, the University of Illinois wunderkind who created the pioneering Mosaic web browser,[81] had released Mosaic for free for noncommercial use, but major companies were not yet in the business of giving away source code.[82] Many in the industry coveted source code as the crown jewel of high technology businesses and were loath to share it.[83]

Eric Schmidt, Sun's Chief Technology Officer who had assured the Java development team that they would be insulated from the business managers, sat at the center of an impending corporate storm. As he would later describe:

> [t]he conversation that never took place, but that I could feel all around me, was, "Eric, you are violating every principle in the company. You are taking our technology and giving it away to Microsoft and every one of our competitors. How are you going to make money?" . . . What I really believed was that Java could create an architectural franchise. The quickest way was through volume and the quickest way to volume was through the Internet.[84]

Sun invited a select group of programmers to test Java secretly in December 1994.[85] The test revealed that the WORA functionality was a game changer and word of Java's capabilities spread like wildfire throughout the programmer community.[86] Sun officially launched Java in January 1995. The business strategy epiphany came when Marc Andreessen gushed to the SAN JOSE MERCURY NEWS that "[w]hat these guys are doing is undeniably, absolutely new. It's great stuff. There's so much stuff people

---

PROPERTY AND INTEROPERABILITY IN THE GLOBAL SOFTWARE INDUSTRY 308 (1995) (noting that Peter Choy, who headed ACIS, worked for Sun).

81. *See* Robert M. Yeh, *The Public Paid for the Invention: Who Owns It?*, 27 BERKELEY TECH. L.J. 453, 492 n. 235 (2012).

82. *See* Michael Calore, *April 22, 1993: Mosaic Browser Lights Up Web with Color, Creativity*, WIRED (Apr. 22, 2010, 12:00 AM), https://www.wired.com/2010/04/0422mosaic-web-browser/.

83. *See* Eugene A. Feher & Dmitriy S. Andreyev, *Source Code Discovery in Patent Litigation*, LAW360 (Apr. 30, 2008), http://www.law360.com/articles/54750/source-code-discovery-in-patent-litigation (noting that "most companies consider their source code to be highly confidential and part of the 'crown jewels' of the company" and that "[s]ource code frequently contains secret proprietary algorithms that provide a vital competitive advantage").

84. *See* Bank, *supra* note 70.

85. *See id.*; Lambert, *supra* note 72, at 139.

86. *See* Bank, *supra* note 70 (reporting that release of early versions of Java in December 1994 "unleashed stratospheric expectations").

want to do over the network that they haven't had the software to do. These guys are really pushing the envelope."[87]

Having already released Java to a select programmer audience, Sun decided to focus on establishing Java as the standard language for web development and figure out how to make money later. It followed the "'profitless' approach to building market share" that Netscape had employed in giving away its Navigator browser.[88]

Due in part to the robust performance of its hardware divisions,[89] Sun could afford to take risks with the revenue side of its software business. Its larger concern, as manifest in the years ahead, was to prevent Microsoft from dominating the emerging Internet marketplace as it had dominated desktop computing software.[90] The WORA approach could be a game changer across the software competition landscape.[91]

In May 1995, Netscape licensed Java as part of its market–leading browser, Navigator.[92] Although Sun authorized Netscape's use for a pittance,[93] it foresaw that this move would rapidly diffuse Java across the programming community and the Web. Sun also provided Java for free to noncommercial users.[94] Java's ability to transform static web pages into engaging, animated, interactive websites revolutionized web design in a matter of months.[95]

Sun actively disseminated Java through low–cost licensing while seeking to prevent fragmentation of the Java platform.[96] Sun's strategy

87. *See* David Bank, *Why Sun Thinks Hot Java Will Give You a Lift New Software Designed to Make World Wide Web's 'Home Pages' More Useful; And Spur Computer Sales*, SAN JOSE MERCURY NEWS, Mar. 23, 1995, at 1A; Bank, *supra* note 70 (quoting Kim Polese, Java's senior product manager: "That quote was a blessing from the god of the Internet.").

88. *See* Bank, *supra* note 70.

89. *See id.* (reporting that Sun's annual revenues from its hardware products were expected to exceed $6 billion in 1995).

90. *See id.* (noting Sun co–founder and CEO Scott McNealy's rivalry with Bill Gates).

91. *See* Mark A. Lemley & David McGowan, *Could Java Change Everything? The Competitive Propriety of a Proprietary Standard*, 43 ANTITRUST BULL. 715 (1998).

92. *See* Lambert, *supra* note 72, at 139.

93. *See* Bank, *supra* note 70 (reporting that Netscape "paid a paltry US$750,000" to license without any per–copy charges).

94. *See id.*

95. *See* Lambert, *supra* note 72, at 139.

96. Sun sued Microsoft over its efforts to undermine the WORA principle. *See* Menell, *supra* note 8; John Markoff, *Sun Sues Microsoft in Dispute Over Java*, N.Y. TIMES (Oct. 8, 1997), https://partners.nytimes.com/library/cyber/week/100897java.html. After

succeeded in establishing Java as the de facto website programming standard.[97] Its open and low licensing cost strategy, however, meant Java was "unlikely ever to become a major profit center at Sun, though any increase in Web traffic is bound to increase sales of Sun's workstations and servers."[98] The success of Sun's hardware division through the 1990s alleviated the need for Sun to earn direct revenues from its Java division.

In 1998, Sun released the Java 2 Standard Edition Platform. It contained eight API packages, three of which—java.lang, java.io, and java.util—were necessary to use the Java programming language.[99] It also established the Java Community Process (JCP), a collaborative mechanism for Java users (including many of the leading software and hardware companies) to expand and update the Java platform.[100]

Over the ensuing years, Sun rolled out updates, improvements, and extensions. Among the goals of the JCP was to bring order to the emerging, but fragmented, mobile device ecosystem. The mobile marketplace was taking off in the mid–1990s with a variety of personal digital assistants

four years of tumultuous litigation Sun and Microsoft settled their litigation in January 2001. *See* Sun Microsystems, Inc. v. Microsoft Corp., 21 F. Supp. 2d 1109 (N.D. Cal. 1998) (granting preliminary injunction enjoining Microsoft from distributing any software implementing Java), *vacated*, 188 F.3d 115 (9th Cir. 1999), *reinstating injunction*, 87 F. Supp. 992 (N.D. Cal. 2000); *see also* Stephen Shankland, *Sun, Microsoft Settle Java Suit*, CNET (Mar. 15, 2002, 5:10 AM), http://www.cnet.com/news/sun-microsoft-settle-java-suit/. Microsoft agreed to pay Sun $20 million and was permanently prohibited from using "Java compatible" trademarks on its products. Sun would later prevail in a separate antitrust and patent infringement action against Microsoft resulting in an award of over $1 billion. *See* Scarlet Pruitt & Paul Roberts, *Microsoft to Pay $700 Million for Antitrust Issues, $900 Million to Resolve Patent Dispute*, INFOWORLD (Apr. 2, 2004), http://www.infoworld.com/article/2667124/operating-systems/update--sun--microsoft-settle-suit-in-billion-dollar-pact.html.

97. *See* Menell, *supra* note 8.

98. *See* Bank, *supra* note 70.

99. *See* Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1349 (Fed. Cir. 2014) [hereinafter *Oracle II*].

100. *See* Simon Ritter, *Keeping the Community in the Java Community Process (JCP)*, SITEPOINT (Nov. 21, 2016); *Java Community Process*, WIKIPEDIA, https://en.wikipedia.org/wiki/Java_Community_Process (last visited Oct. 8, 2017). Sun also sought to have Java recognized as a de jure standard programming language for Internet through the International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC). Sun ultimately withdrew its application as opposition mounted to a formal public standard largely controlled by a single company. *See* Lemley & McGowan, *supra* note 91, at 755.

(PDAs), cell phones, and other consumer devices. In 1998 and 1999, Sun worked with JCP members to develop the Java 2 Micro Edition (J2ME).[101]

Sun's hardware sales collapsed when the dot–com bubble burst in early 2000 as many of the companies that had ordered Sun hardware went bankrupt. Sun's stock went into freefall. As the technology sector recovered in 2004, advanced microcomputers displaced demand for higher–end Sun workstations. Sun canceled major processor projects, closed one of its two major factories, and initiated a series of layoffs.[102]

Sun came to see Java and software as its future. To expand Java's reach, Sun licensed Java Standard Edition, Enterprise Edition, and Micro Edition under the GNU GPLv2 in 2006.[103] Symbolizing its shift in direction, Sun changed its NASDAQ ticker in August 2007 from SUNW to JAVA.[104]

Sun's struggles continued, however, resulting in further deep losses during the 2008 financial crisis. Its market value fell 80% between November 2007 and November 2008, resulting in further layoffs.[105]

2. The Android Platform

Drawing on the Navigator and Java strategy, Google focused on widespread adoption rather than revenue for its eponymous search engine. It offered free access. As the technology press recognized its "uncanny knack for returning extremely relevant results,"[106] Google amassed loyal users and separated itself from the crowded pack of search engines. Unlike Netscape and Sun, however, Google developed a robust revenue model for its free–to–users software: keyword advertising. By October 2000, just as Sun's hardware business was setting, Google's AdWords program was

101. *See* Ritter, *supra* note 100; *J2ME Programming/The J2ME Platform*, WIKIBOOKS, https://en.wikibooks.org/wiki/J2ME_Programming/The_J2ME_Platform (last visited Oct. 8, 2017).

102. *See* Menell, *supra* note 8.

103. *See* Steven J. Vaughan-Nichols, *Sun to Open-Source Java under GPL*, PRACTICAL TECH. (Nov. 11, 2006), http://practical-tech.com/development/sun-to-open-source-java-under-gpl/415/. The GNU GPL license requires that software built on the open source code base be available to others on an open source basis—the so–called share–alike requirement. *See* Brian C. Carver, *Share and Share Alike: Understanding and Enforcing Open Source and Free Software Licenses*, 20 BERKELEY TECH. L.J. 443, 455 (2005).

104. *See Sun Microsystems' New Ticker: JAVA*, L.A. TIMES (Aug. 24, 2007), http://articles.latimes.com/2007/aug/24/business/fi-wrap24.s4.

105. *See* Devlin, *supra* note 73; Ashlee Vance, *Sun Microsystems Reports $1.7 Billion Loss and Falling Sales*, N.Y. TIMES (Oct. 30, 2008), http://www.nytimes.com/2008/10/31/technology/companies/31sun.html.

106. *See* Don Willmot, *Top 100 Web Sites*, PC MAG., Feb. 9, 1999, at 118.

launched.[107] In August 2001, Google named Eric Schmidt, Sun's former CTO as its Chief Executive Officer. The press release touted that Schmidt had "led the development of Java, Sun's platform-independent programming technology, and defined Sun's Internet software strategy."[108]

With revenue flowing from AdWords, Google developed a series of new projects—image search, news, shopping, Gmail, maps—which reinforced and expanded its advertising business. Google went public in 2004[109] and continued to expand its reach with Google Books, YouTube, and other projects.[110]

Google's leaders foresaw mobile devices as a substantial risk to their advertising juggernaut.[111] The early smartphones, such as RIM's BlackBerry, did not make effective use of Google's advertising links.[112]

---

107. *See* GOOGLE CO., *Our History in Depth*, https://www.google.com/about/company/history (last visited Oct. 8, 2017); Jemima Kiss, *Ten Years of Online Advertising with Google Adwords*, GUARDIAN (Oct. 25, 2010, 2:00 PM), https://www.theguardian.com/media/2010/oct/25/advertising-google-adwords.

108. *See Google Names Dr. Eric Schmidt Chief Executive Officer*, GOOGLE (Aug. 6, 2001), www.googlepress.blogspot.com/2001/08/google-names-dr-eric-schmidt-chief.html.

109. *See* John Markoff, *THE GOOGLE I.P.O.: THE OVERVIEW; Google's Sale of Its Shares Will Defy Wall St. Tradition*, N.Y. TIMES (Apr. 30, 2004), http://www.nytimes.com/2004/04/30/business/google-ipo-overview-google-s-sale-its-shares-will-defy-wall-st-tradition.html.

110. *See* Stephen Shankland, *Google Takes Bold Action to Match Aspirations*, CNET (Aug. 15, 2011, 10:54 AM), https://www.cnet.com/news/google-takes-bold-action-to-match-aspirations/.

111. In its 2005 10-K filing, Google identified the emerging mobile marketplace as a potential threat to its profitability:

> ***More individuals are using non-PC devices to access the Internet, and versions of our web search technology developed for these devices may not be widely adopted by users of these devices.***
>
> The number of people who access the Internet through devices other than personal computers, including mobile telephones, hand-held calendaring and e-mail assistants, and television set-top devices, has increased dramatically in the past few years. The lower resolution, functionality and memory associated with alternative devices make the use of our products and services through such devices difficult. If we are unable to attract and retain a substantial number of alternative device users to our web search services or if we are slow to develop products and technologies that are more compatible with non-PC communications devices, we will fail to capture a significant share of an increasingly important portion of the market for online services.

Google Inc., Annual Report 32 (Form 10-K) (Dec. 31, 2005).

112. *See* FRED VOGELSTEIN, DOGFIGHT: HOW APPLE AND GOOGLE WENT TO WAR AND STARTED A REVOLUTION 53 (2013).

Developing a new mobile platform, however, posed daunting challenges. The mobile marketplace was a morass of telecommunication companies, handset makers, and software providers.[113] The telecommunications companies (telcos) jealously guarded their networks.[114] The handset makers (original equipment manufacturers (OEMs)) had divergent strategies and business models. Microsoft and Symbian were promoting proprietary mobile operating systems but without notable success. Google executives feared that Microsoft could steer consumers away from Google search if Microsoft successfully established a mobile platform.[115]

Google saw promise in Android, a startup founded in October 2003 to develop "smarter mobile devices that are more aware of [their] owner's location and preferences."[116] Android's founder, Andy Rubin, had previously developed T–Mobile's Sidekick, a compact mobile device that provided an authentic web browsing experience.[117] Rubin and Google recognized that an open, competitive strategy could potentially overcome the structural factors impeding development of a breakthrough mobile platform.[118] Google acquired Android for $50 million in July 2005 and put Rubin in charge of its new mobile division.[119]

At the first high-level Android planning meeting, the newly established Android team and Google leaders focused on three questions:

> • Which type of Open Source are we?
>
> • How do we interact with the OSS [open source software community]?
>
> • How do we Open Source our JVM [Java Virtual Machine]?[120]

113. *See id.* at 48–50.

114. *See* John Markoff, *I, Robot: The Man Behind the Google Phone*, N.Y. TIMES (Nov. 4, 2007), http://www.nytimes.com/2007/11/04/technology/04google.html.

115. *See* VOGELSTEIN, *supra* note 112, at 51.

116. *See* Ben Elgin, *Google Buys Android for Its Mobile Arsenal*, BUS. WEEK (Aug. 17, 2005), http://tech-insider.org/mobile/research/2005/0817.html.

117. *See* John Markoff, *Where Does Google Plan to Spend $4 Billion?*, N.Y. TIMES (Aug. 22, 2005), http://www.nytimes.com/2005/08/22/technology/where-does-google-plan-tospend-4-billion.html (observing that Page and Brin wore the Sidekick all–purpose voice and data communicators on their belts several years ago and that Page had long envisioned a Google branded smart phone); VOGELSTEIN, *supra* note 112, at 52–53.

118. *See* VOGELSTEIN, *supra* note 112, at 49.

119. *See* Markoff, *supra* note 117.

120. *See* Trial Ex. 1, Android GPS [Google Product Strategy]: Key strategic decisions around Open Source at 2, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (3:10-cv-03561-WHA) (filed July 26, 2005) [hereinafter Trial Exhibit 1].

The Android team planned to use a permissive open source license that merely required licensees to maintain compatibility with Google APIs.[121] Several factors made Java a critical part of their plan: the carriers required it; Microsoft would never pursue it; Java had well–developed, tested tools; Java assured third party app developers that the platform would remain available; a large and growing pool of developers knew Java; and handset makers could cheaply license Java from Sun.[122]

Sun's Java 2 Mobile Edition, which was widely used on feature phones,[123] would not be adequate for the Android platform for several reasons. First, Google sought to design a new platform optimized for the small chips on handsets and add new functionalities.[124] It aimed to use some of the Java API packages and develop others. Second, Google sought to use a less restrictive licensing model than the GNU GPL in order to promote robust downstream innovation and competition.[125] The GPL's viral "share and share alike" provision would prevent handset makers and telephone companies from commercializing proprietary extensions on top of the base platform.[126] Furthermore, these vendors would not want to share that technology under the viral licensing model.[127]

The Android team believed that they could achieve their goals by negotiating the first open source Java 2 Platform, Micro Edition license with Sun.[128] The preliminary negotiations went well, and both sides believed

121. *See id*.

122. *See id*. at 8.

123. The term "feature phone" characterizes low–end mobile phones with limited capability–principally voice and text messaging with basic multimedia and rudimentary internet access. They have relatively small screens. *See* Nicole Lee, *The 411: Feature Phones vs. Smartphones*, CNET (Mar. 1, 2010, 5:14 PM) https://www.cnet.com/news/the-411-feature-phones-vs-smartphones/.

124. *See* Trial Exhibit 1, *supra* note 120, at 2.

125. *See id*.

126. *See id*.

127. *See* Trial Ex. 230, email from Andy Rubin to Bob Lee (Aug. 11, 2007), Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (3:10-cv-03561-WHA) ("The problem with GPL in embedded systems is that it's viral, and there is no way (for example) OEMs or Carriers to differentiate by adding proprietary works. We are building a platform where the entire purpose is to let people differentiate on top of it.").

128. *See* Trial Exhibit 1, *supra* note 120, at 2. The memo noted that Tim Lindholm, a former Sun Microsystems engineer who was involved with Java, and who then worked for Google, would lead the negotiation. *See id.* at 9; John Letzing, *Who Is Tim Lindholm? Google's CEO is Wondering That Too*, WALL ST. J. (Apr. 18, 2012), http://blogs.wsj.com/digits/2012/04/18/who-is-tim-lindholm-googles-ceo-is-wondering-that-too/.

they could reach an agreement.[129] The negotiations unraveled, however, over Google's unwillingness to agree to make Android fully compatible with the Java platform.[130] Sun demanded strict adherence to the WORA principle, which was a deal–breaker for Google.

Google pushed ahead with its selective use of Java API packages by independently implementing the functional specifications in a clean–room environment.[131] Using the Java language would not be a problem as it had long been freely available. But the Android team also wanted to use selected Java API packages from the Java™ Standard Edition and develop its own virtual machine. If the Java programming language is analogized to the letters, words, and syntax of the English language, the API implementations can roughly be characterized as paragraphs or chapters within a book. Copying the full API implementations—involving large chunks of source code—would run afoul of copyright law. Android could achieve its goals by emulating the API functionality with independently written implementing code. And by avoiding restrictive licensing terms with Sun, Google could blaze its own trail free of Sun's interference.[132]

---

129. *See* Trial Ex. 13, email from Brian Swetland to Mathias Agopian (Jan. 2, 2006), Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (3:10-cv-03561-WHA); Doc. 398-10, email from Andy Rubin to Sergey Brin, (Jan. 13, 2006), Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (3:10-cv-03561-WHA), http://www.fosspatents.com/2011/09/sun-proposed-red-hat-style-android.html; Trial Ex. 16, email from Scott McNealy to Vineet Gupta (Feb. 9, 2006), Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (3:10-cv-03561-WHA); Trial Ex. 2372, email from Vineet Gupta to Andy Rubin (May. 8, 2006), Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (3:10-cv-03561-WHA).

130. *See* Trial Ex. 214, email from Eric Schmidt to Andy Rubin (May 14, 2006), Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (3:10-cv-03561-WHA); Trial Ex. 215, email from Chris Desalvo to Andy Rubin (June 1, 2006), Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (3:10-cv-03561-WHA); VOGELSTEIN, *supra* note 112, at 57 (reporting that Sun would not agree to forking of its platform); *see also* Trial Ex. 230, email from Andy Rubin to Bob Lee (Aug. 11, 2007), Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (3:10-cv-03561-WHA) (explaining Sun's profit motivation for choosing GPL for Java ME: "Sun chose GPL . . . so that companies would need to come back to them and take a direct license and pay royalties," and noting that Google "negotiated 9 months with Sun and decided to walk away after they threatened to sue us over patent violations.").

131. *See* Trial Ex. 215, email from Chris Desalvo to Andy Rubin (June 1, 2006), Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (3:10-cv-03561-WHA) ("With talks with Sun broken off where does that leave us regarding Java class libraries? Ours are half-ass at best. We need another half of an ass.").

132. *See* Trial Ex. 18, email from Andy Rubin to Greg Stein (Mar. 24, 2006), Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (3:10-cv-03561-WHA) (expressing consternation at Sun's licensing model: "Ha, wish them luck. Java.lang api's

Google recognized that this path risked copyright and patent infringement. The copyright issue turned on whether and to what extent copyright law protected the function labels and structure, sequence, and organization (SSO) of Java APIs. As a result of the Supreme Court's deadlock, the *Lotus v. Borland* decision, which cleared the way for copying of function labels, strictly governed only the First Circuit. The Second Circuit's *Altai* decision and the Ninth Circuit's *Apple* decision exposed the weakness of the Third Circuit's superficial analysis of SSO in *Whelan*. Furthermore, the *Altai* decision and the Ninth Circuit's *Sega* decision clearly viewed achieving interoperability with another computer interface through a different implementation to be fair game, but Android was aiming for something other than end–user interoperability. It wanted to pick and choose among interface elements in building a new platform—an optimized interface for a different consumer product.

Over the next two years, the Android team independently implemented 37 of the 166 Java API packages in the Java™ Standard Edition[133] and developed an independent virtual machine ("Dalvik"). In this way, the Android operating system emulated the functionality of known and tested APIs that fit the Android team's constrained design parameters. Android's use of the same function labels as Java would enable millions of Java programmers to quickly master Android app development. Although Android apps would not be fully interoperable with Java, they would be similar and better optimized to the constraints of mobile devices.[134] This clean–room effort added substantially more time and cost to Android's development, but avoided literal copying of the Java API implementation code.[135]

With the breakthrough success of the Apple iPhone in 2007, Google came to see Android as critical to its business strategy.[136] The iPhone propelled Apple into a dominant position in the emerging smartphone marketplace. Google feared that Apple could rule mobile technology in

---

are copyrighted. And Sun gets to say who they license the tck [Technology Compatibility Kit used to ensure Java compatibility] to, and forces you to take the 'shared part' which taints any clean room implementation.").

133. *See* Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974, 977 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014) [hereinafter *Oracle I*].

134. *See* Stephen Shankland, *Google Carves an Android Path Through Open-source World*, CNET (May 22, 2008), http://www.cnet.com/news/google-carves-an-android-path-through-open-source-world/.

135. *See* VOGELSTEIN, *supra* note 112, at 57.

136. *See id.* at 129–30.

much the same way that Microsoft had ruled the desktop market, thereby threatening Google's strength in search and other Internet services. In response, Google allocated more resources to the Android project.[137] The Android team found it far easier to negotiate partnerships with the many telcos and handset manufacturers marginalized by Apple's decision to produce its own device and license the iPhone exclusively with AT&T.[138]

Google began the rollout of the Android platform in early November 2007.[139] On November 5th, Google unveiled the Open Handset Alliance, a consortium of handset makers, application developers, telcos, and components manufacturers (such as chip makers), in conjunction with the outlines of the Android platform.[140] Andy Rubin explained that Android's software was based on the Linux operating system and Sun's Java language, which would enable programmers to easily develop applications that connect to independent Web services.[141]

Jonathan Schwartz, Sun's CEO, publicly applauded Google's use of Java, proclaiming that Google had "strapped another set of rockets to the [Java] community's momentum—and to the vision-defining opportunity across our (and other) planets."[142] Privately, Sun feared that Android's use of Java would undermine its WORA paradigm and its mission to establish Java ME as the leading mobile platform and a significant revenue generator.[143]

---

137. *See* Trial Ex. 433, Android GPS Meeting Notes (July 17, 2007), Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (3:10-cv-03561-WHA); VOGELSTEIN, *supra* note 112, at 83–84.

138. *See* VOGELSTEIN, *supra* note 112, at 119–21.

139. *See Industry Leaders Announce Open Platform for Mobile Devices: Group Pledges to Unleash Innovation for Mobile Users Worldwide*, OPEN HANDSET ALLIANCE (Nov. 5, 2007), http://www.openhandsetalliance.com/press_110507.html; Miguel Helft & John Markoff, *Google Enters the Wireless World*, N.Y. TIMES (Nov. 5, 2007), http://www.nytimes.com/2007/11/05/technology/05cnd-gphone.html; Saul Hansell, *The Gphone: So Open It Could Be Closed*, N.Y. TIMES: BITS BLOG (Nov. 5, 2007), http://bits.blogs.nytimes.com/2007/11/05/the-gphone-so-open-it-could-be-closed/.

140. *See* Erick Schonfeld, *Breaking: Google Announces Android and Open Handset Alliance*, TECHCRUNCH (Nov. 5, 2007), https://techcrunch.com/2007/11/05/breaking-google-announces-android-and-open-handset-alliance/.

141. *See* Helft & Markoff, *supra* note 139.

142. *See* Jay Greene, *Scoop: Oracle Scrubs Site of Embarrassing Java Blog*, CNET (July 22, 2011, 10:09 PM), https://www.cnet.com/uk/news/scoop-oracle-scrubs-site-of-embarrassing-java-blog/.

143. *See* Trial Ex. 565, email thread from Vineet Gupta (Sep. 24, 2007), Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (3:10-cv-03561-WHA). In an "off-the-record" communication with a New York Times reporter one day after the Android announcement, Schwartz sniped about Google's opposition to Sun's plan to open

The following week, Google released the "open source" Android Software Development Kit (SDK), which enabled companies to build their own smartphones.[144] Google did not, however, release the Android source code, indicating that it would not be available until the first Android phones went on sale in late 2008.[145] Nor did Google release its own branded phone, although it left that option open.

Based on the Android SDK, Sun and other industry observers could see that Google was diverging from the Java standard platform and the Java Community Process.[146] Google deflected suggestions that Android fragmented Java by focusing attention on how the Open Handset Alliance provides a more responsive, less restrictive, open platform for mobile devices.[147] Sun and Google continued to monitor each other's activities warily as Android products moved into the marketplace in 2008 and

source Java. *See* Trial Ex. 2371, email from Jonathan Schwartz to John Markoff (Nov. 6, 2007), Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (3:10-cv-03561-WHA), http://www.fosspatents.com/2012/04/former-sun-chief-about-google-immune-to.html.

144. *See* Steve Horowitz, *Calling All Developers: $10M Android Challenge*, GOOGLE OFFICIAL BLOG (Nov. 12, 2007), https://googleblog.blogspot.com/2007/11/calling-all-developers-10m-android.html ("Today, the team is releasing an early look at the Android SDK for developers interested in building applications for Android. To get things rolling, we've also announced the Android Developer Challenge, which provides $10 million in awards for developers who build great applications for Android.").

145. *See* Peter Judge, *Google Android on the Defensive: Google Defends Decision to Use its Own Flavor of Java in the Android SDK Rather Than Support the Popular C++*, INFOWORLD (Nov. 15, 2007), http://www.infoworld.com/article/2651252/networking/google-android-on-the-defensive.html.

146. *See* Stephen Shankland, *Sun's Worried That Google Android Could Fracture Java*, CNET (Nov. 14, 2007), http://www.cnet.com/news/suns-worried-that-google-android-could-fracture-java/ ("Painful flashbacks are beginning to torment those of us who lived through the Java wars between Sun Microsystems and Microsoft that began 10 years ago. Earlier this week, Google released programming tools for its Android mobile-phone software project that shun the existing Java standard-setting process in favor of a Google-specific variety. Sun responded on Wednesday by expressing concern that Google's Android project could fragment Java into incompatible versions."); *see also* Stephen Shankland, *Google's Android Parts Ways with Java Industry Group*, CNET (Nov. 13, 2007), http://www.cnet.com/news/googles-android-parts-ways-with-java-industry-group/.

147. *See* Stephen Shankland, *Sun's Worried That Google Android Could Fracture*, CNET (Nov. 14, 2007), http://www.cnet.com/news/suns-worried-that-google-android-could-fracture-java/ (quoting a Google press statement: "Google and the other members of the Open Handset Alliance are working to help solve fragmentation and supporting the developer community by creating Android, a mobile platform that responds to the needs of the developers, has the backing of industry leaders, and will be available as open source under a nonrestrictive license.").

2009,[148] a period in which Apple's iPhone was ascendant. Leaders at both companies occasionally broached licensing and collaboration,[149] but a gulf remained.[150] Sun refrained from blocking Android through legal action.

The marketplace resolved the fate of the two companies. Rubin's vision proved prescient: "When you have multiple [handset makers] building multiple products in multiple product categories, it's just a matter of time before sales of Android phones exceed the sales of proprietary systems like Apple's and R.I.M.'s."[151] After a gradual start, Android took the global smartphone operating systems market by storm, surpassing the Apple iOS

148. *See* Trial Ex. 2070, email from Vineet Gupta to Jonathan Schwartz (Oct. 23, 2008), Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (3:10-cv-03561-WHA) (indicating that Google's Android "proposal more than likely is going to be about buying out Java"); Trial Ex. 29, email from Rubin to Dick Wall (Mar. 24, 2008), Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (3:10-cv-03561-WHA) (warning Google representatives not to demonstrate Android features to Sun employees or lawyers at JavaOne convention), Trial Ex. 326, email from Dave Sobota to Tim Lindholm (Feb. 19, 2009), Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (3:10-cv-03561-WHA) (raising the question of who will own Java if Sun collapses and suggesting Google could buy the patent and copyright rights as a way of making "[o]ur Java lawsuits go away"); Trial Ex. 1029, email from Tim Lindholm to Dan Bornstein (Apr. 29, 2009), 872 F. Supp. 2d 974 (N.D. Cal. 2012) (3:10-cv-03561-WHA) (recommending avoiding interaction with Sun to avoid "inadvertently stir[ring] anything up for Android").

149. *See* Trial Ex. 1002, email thread from Tim Lindholm to Andy Rubin (Nov. 24, 2008), 872 F. Supp. 2d 974 (N.D. Cal. 2012) (3:10-cv-03561-WHA) (discussing recent efforts by Sun to "certify Android through the Java process and become licensees of Java"); Trial Ex. 3466, email from Eric Schmidt to Jonathan Schwartz (Mar. 31, 2008), 872 F. Supp. 2d 974 (N.D. Cal. 2012) (3:10-cv-03561-WHA) (Re: update on android licensing; "We are happy to have our team meet with anyone at Sun who would like more information or who has ideas for us"; calling attention to an explanation of why Google chose to distribute Android to the public using the Apache v2 license); *see also* Ryan Paul, *Why Google Chose the Apache Software License over GPLv2 for Android*, ARS TECHNICA (Nov. 6, 2007), http://arstechnica.com/uncategorized/2007/11/why-google-chose-the-apache-software-license-over-gplv2/ (linked in Eric Schmidt's March 31, 2008 email to Jonathan Schwartz).

150. Sun had proposed to license Java to Google for $60 million over three years plus an additional amount of up to $25 million per year in revenue sharing. *See* Doc. 182, Letter from Scott Weingaertner, Counsel, Google, to Judge William Alsup at 5 (June 6, 2011) 872 F. Supp. 2d 974 (N.D. Cal. 2012) (3:10-cv-03561-WHA), https://www.scribd.com/document/58133136/Oracle-Google-Damages-June-6-Precis-Unredacted. It is unclear whether that offer would have afforded Google the flexibility and independence in developing Android that it sought.

151. *See* Brad Stone, *Google's Andy Rubin on Everything Android*, N.Y. TIMES: BITS BLOG (Apr. 27, 2010), http://bits.blogs.nytimes.com/2010/04/27/googles-andy-rubin-on-everything-android/ (internal quotation marks omitted).

market share by mid–2010 and leaving Java ME, RIM, Microsoft, and Symbian in the dust.[152]

With its hardware business in decline, software acquisitions sputtering,[153] and ability to monetize Java diminished, Sun Microsystems' ability to survive as an independent company came into question.[154] Oracle Corporation, one of the strongest software companies that had built many of its products on the Java platform, swooped in.[155]

Oracle's acquisition of Sun brought legal action against Google into play. Notwithstanding consternation over Android's "unofficial," nonstandard, and incomplete Java implementation,[156] suing Google would have gone against Sun's long–standing cultural norms about open technology and evangelism within the industry.[157] Moreover, Sun could ill afford a prolonged litigation battle or the risk to Sun's reputation with other technology companies. Google was well–positioned financially and legally to put up a stiff defense. Sun's business was struggling, and Wall Street and

152. *See Global Mobile OS Market Share in Sales to End Users From 1st Quarter 2009 to 1st Quarter 2017*, STATISTA (2017), http://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/.

153. Sun had purchased StorageTek, a storage vendor, for $4.1 billion in 2005 and MySQL, a relational database company, for $1 billion in 2008. *See* Jon Brodkin, *The Downfall of Sun Microsystems*, NETWORKWORLD (Apr. 24, 2009), http://www.networkworld.com/article/2268096/servers/the-downfall-of-sun-microsystems.html.

154. *Id.*

155. *See* Patrick Thibodeau & Elizabeth Montalbano, *Update: Oracle Buying Sun in $7.4B Deal*, COMPUTERWORLD (Apr. 20, 2009), http://www.computerworld.com/article/2523479/data-center/update--oracle-buying-sun-in--7-4b-deal.html.

156. *See* Dan Farber, *Java Creator James Gosling: 'Google Totally Slimed Sun'*, CNET (Apr. 30, 2012), http://www.cnet.com/news/java-creator-james-gosling-google-totally-slimed-sun/ (quoting Gosling stating that Sun was "wronged" by Google and citing Sun's objections to Android's "very weak notions of interoperability" with Java); Gavin Clarke, *Ellison Wrestles Google to Strangle 'Unofficial' Java*, REGISTER (Aug. 13, 2010), (referring to Android as an "unofficial" Java software platform), https://www.theregister.co.uk/2010/08/13/oracle_google_java_prosecution/; Joe Mullin, *Sun's Jonathan Schwartz at Trial: Java Was Free, Android Had No Licensing Problem*, ARS TECHNICA (May 11, 2016), http://arstechnica.com/tech-policy/2016/05/suns-jonathan-schwartz-at-trial-java-was-free-android-had-no-licensing-problem/ (quoting former Sun CEO expressing annoyance at Google's refusal to work out a license with Sun).

157. *See* Brad Feld, *Oracle's Java API Suit Against Google—Five Years Later*, FELDTHOUGHTS (June 29, 2015), http://www.feld.com/archives/2015/06/oracles-java-api-suit-google-five-years-later.html (observing that "[f]iling patent suits was never in Sun's genetic code" (quoting James Gosling, *The Shit Finally Hits the Fan....*, JAVAVIRTUALMACHINE.NET (Aug. 13, 2010), http://news.java-virtual-machine.net/6018.html)); Mullin, *supra* note 156.

potential suitors would likely have seen such a lawsuit as a sign of desperation and a distraction from Sun's business goals.

Oracle's acquisition of Sun Microsystems dramatically altered the Java enforcement equation. Larry Ellison, Oracle's cofounder and CEO, had a reputation for brash business tactics. Whereas Sun's leadership had embraced open technology with religious fervor, Oracle's approach had been strategic. Unlike Sun, Oracle possessed the financial strength and diversified business strategy to pursue high stakes litigation. It had done well in recent years pursuing corporate takeovers and copyright litigation against SAP.[158]

In announcing the Sun acquisition, Ellison characterized Java as "the single most important software asset we have ever acquired" and touted Oracle's Java–based, middleware business, bolstered first by its BEA Systems acquisition[159] and purchase of Sun, as being "on track to become as large as Oracle's flagship database business."[160] Oracle would need to reposition Java's licensing business to achieve that goal. Oracle's leadership team sought to pursue a far more aggressive Java licensing strategy. It believed that Sun products could bring in $1.5 billion in operating profits in the first year following the acquisition.[161]

Oracle completed its acquisition of Sun in early 2010.[162] Oracle immediately approached Google about its use of Java in the Android

158. *See* Jim Henschen, *Oracle Lawsuit Against SAP Settled at Law*, INFORMATIONWEEK (Nov. 14, 2016, 8:50 AM), http://www.informationweek.com/cloud/software-as-a-service/oracle-lawsuit-against-sap-settled-at-last/d/d-id/1317483; Verne F. Kopytoff, *SAP Ordered to Pay Oracle $1.3 Billion*, N.Y. TIMES (Nov. 23, 2010), http://www.nytimes.com/2010/11/24/business/24oracle.html; Danny Diegal, *Oracle Takes Reduced $357M Award in SAP Copyright Suit*, LAW360 (Nov. 13, 2016, 9:42 PM), https://www.law360.com/articles/596188/oracle-takes-reduced-357m-award-in-sap-copyright-suit; Steve Lohr & Laurie J. Flynn, *Oracle to Acquire PeopleSoft for $10.3 Billion, Ending Bitter Fight*, N.Y. TIMES (Dec. 14, 2004), http://www.nytimes.com/2004/12/14/technology/oracle-to-acquire-peoplesoft-for-103-billion-ending-bitter-fight.html.

159. *See* Larry Dugan, *Surprise! Oracle Buys BEA Systems*, ZDNET (Jan. 16, 2008), http://www.zdnet.com/article/surprise-oracle-buys-bea-systems/. BEA Systems specialized in enterprise infrastructure software products. *See id.*

160. *See* Patrick Thibodeau & Elizabeth Montalbano, *Update: Oracle Buying Sun in $7.4B Deal*, COMPUTERWORLD (Apr. 20, 2009), http://www.computerworld.com/article/2523479/data-center/update--oracle-buying-sun-in--7-4b-deal.html.

161. *See* Brodkin, *supra* note 153.

162. Antitrust authorities in the U.S. and Europe delayed the acquisition out of concern that Oracle—the leading relational database vendor—was acquiring a promising competing business (MySQL). *See* James Kanter, *New Snag for Oracle in Sun Deal*, N.Y.

platform. Google considered alternatives to Java,[163] but ultimately stood its ground due to the lack of adequate workarounds. For Oracle, the prospect of spending millions of dollars on attorneys' fees for even a modest possibility of sharing in the large and growing Android marketplace was a plausible, if not attractive, business proposition. Moreover, it could quickly establish Oracle as a key player in the lucrative, strategically important, and rapidly growing mobile operating system marketplace. Delay would only enhance Google's laches and equitable–estoppel defenses.

C. THE *ORACLE V. GOOGLE* LITIGATION

After six months of negotiations with Google, Oracle filed a broadside salvo alleging Android infringed Java–related patents and copyrights in the Northern District of California.[164] With billions of dollars and control of two of the most important software platforms at stake, the parties spared no expense in litigating the case over the next six years.

The case was assigned to Judge William Alsup, a respected jurist who was unafraid to grapple with complex technologies.[165] Judge Alsup actively managed the case, pushing the parties to settle the dispute or move to trial quickly. He pressured Oracle to streamline its patent allegations[166] and

---

TIMES (Sept. 3, 2009), http://www.nytimes.com/2009/09/04/technology/companies/04oracle.html.

163. *See* Florian Mueller, *Google's Five Failed Attempts to Give Confidential Status to 'Damning' E-mail in Oracle Case*, FOSS PATENTS (Nov. 9, 2011), http://www.fosspatents.com/2011/11/googles-five-failed-attempts-to-give.html (noting that Page and Brin had asked engineers "to investigate what technical alternatives exist to Java for Android and Chrome. We have been over a bunch of these, and think they all suck. We conclude that we need to negotiate a license for Java under the terms we need." (quoting Trial Ex. 10, email from Tim Lindholm to Andy Rubin et al. (Aug. 6, 2010), Oracle Am., Inc. v. Google Inc. 872 F. Supp. 2d 974 (N.D. Cal. 2012) (3:10-cv-03561-WHA))).

164. *See* Complaint for Patent and Copyright Infringement, Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (3:10-cv-03561-WHA), 2010 WL 11221486.

165. *See* Dan Farber, *Judge William Alsup: Master of the Court and Java*, CNET (May 31, 2012), http://www.cnet.com/news/judge-william-alsup-master-of-the-court-and-java/.

166. *See* Florian Mueller, *Oracle-Google Trial to Start on April 16, 2012*, FOSS PATENTS (Mar. 13, 2012), http://www.fosspatents.com/2012/03/oracle-google-trial-to-start-on-april.html; Florian Mueller, *Oracle Offers Withdrawal of Three More Patents in Exchange for Spring Trial Against Google*, FOSS PATENTS (Mar. 9, 2012), http://www.fosspatents.com/2012/03/oracle-offers-withdrawal-of-three-more.html; Florian Mueller, *Pressure Mounting on Oracle to Drop Patent Claims Against Google and Focus on Copyright*, FOSS PATENTS (Mar. 5, 2012), http://www.fosspatents.com/2012/03/pressure-mounting-on-oracle-to-drop.html.

rejected Google's summary judgment motion, which asserted that the API packages were uncopyrightable.[167] While agreeing with Google that "the names of the Java language API files, packages, classes, and methods are not protectable as a matter of law"[168] under the copyright doctrine denying protection for names and short phrases,[169] the court nonetheless rejected Google's broader argument that API declarations (beyond short phrases) and documentation are unprotectable under the *scènes à faire*, merger, or methods of operation (§ 102(b)) doctrines.

1. The 2012 Trial

Judge Alsup structured the trial in three phases: (I) copyright infringement claims; (II) patent infringement claims; and (III) all remaining issues, including damages and willfulness, if necessary.[170] As the case wended its way toward trial, the core copyright allegations were boiled down to: (a) "12 Android files of source code (copied from 11 Java files), including rangeCheck"; (b) "Plain English descriptions in the user manual, sometimes called the API 'specifications'"; (c) "37 APIs but only as to their specific selection, structure, and organization, it being conceded that the implementing code is different"; and (d) "Android's entire source code and object code as derivative works of the 37 Java APIs."[171] The parties agreed that Judge Alsup would decide the copyrightability of the Java APIs and that the jury would decide copyright infringement, fair use, and whether any copying was de minimis.[172] Thus, a jury would not hear the most salient copyright issue the Oracle–Google litigation raised—the copyrightability of APIs.

167. *See* Oracle Am., Inc. v. Google Inc., 810 F. Supp. 2d 1002, 1005 (N.D. Cal. 2011).

168. *Id.* at 1009–10.

169. *See* Material Not Subject to Copyright, 37 C.F.R. 202.1(a) (2016) (regulation denying copyright registration for "[w]ords and short phrases such as names, titles, and slogans"); *see also* Planesi v. Peters, No. 04-16936, 2005 WL 1939885, *1 (9th Cir. Aug. 15, 2005); Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1524 n.7 (9th Cir. 1992) ("Sega's security code is of such *de minimis* length that it is probably unprotected under the words and short phrases doctrine.").

170. *See* Oracle Am., Inc. v. Google Inc., 3:10-cv-03561-WHA, 2012 WL 1189898, (N.D. Cal. Jan. 4, 2012).

171. *See* Request for Statement of Issues Re: Copyright at 1–2, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014) (3:10-cv-03561-WHA), https://docs.justia.com/cases/federal/district-courts/california/candce/3:2010cv03561/231846/854.

172. *Oracle I*, 872 F. Supp. 2d at 975.

As a result of Judge Alsup's decision to reserve the API copyrightability question to himself, the jury's infringement verdict was largely a foregone conclusion. Judge Alsup instructed the jury that Oracle's Java–related copyrights "cover the structure, sequence and organization [SSO] of the compilable code"[173] and that Google "agrees that the structure, sequence and organization of the 37 accused API packages in Android is substantially the same as the structure, sequence and organization of the corresponding 37 API packages in Java."[174] Judge Alsup further instructed the jury that "[w]hile individual names are not protectable on a standalone basis, names must necessarily be used as part of the structure, sequence, and organization and are to that extent protectable by copyright."[175]

Oracle's principal copyright infringement argument boiled down to showing the jury a side–by–side comparison of Java and Android source code. Beyond its motion seeking a determination that the Java APIs are not copyrightable,[176] Google's principal path to a trial victory was that the jury would find that Android's use of Java was permissible under the fair use doctrine.

Oracle secured a partial victory in the copyright phase of the trial.[177] While concluding that Android infringed the 37 Java API packages in question taken as a group,[178] the jury found that Google did not infringe

173. *See* Doc. 1018, Final Charge to the Jury (Phase One) and Special Verdict Form at 8, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014) (3:10-cv-03561-WHA) (filed Apr. 30, 2012), https://docs.justia.com/cases/federal/district-courts/california/candce/3:2010cv03561/231846/1018.

174. *See id*. at 10.

175. *See id*. at 20.

176. *See* Doc. 984, Google's Motion for Judgment as a Matter of Law on Sections of Count VIII of Oracle's Amended Complaint, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014) (3:10-cv-03561-WHA), 2012 WL 3992644.

177. *See* Doc. 1089, Special Verdict Form, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014) (3:10-cv-03561-WHA) (filed May 7, 2012),https://docs.justia.com/cases/federal/district-courts/california/candce/3:2010cv03561/231846/1089; Joe Mullin, *Google Guilty of Infringement in Oracle Trial*, ARS TECHNICA (May 7, 2012), http://arstechnica.com/tech-policy/2012/05/jury-rules-google-violated-copyright-law-google-moves-for-mistrial/.

178. *See* Doc. 1089, Special Verdict Form at 1, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014) (3:10-cv-03561-WHA) (filed May 7, 2012), https://docs.justia.com/cases/federal/district-courts/california/candce/3:2010cv03561/231846/1089.

Java documentation[179] and that the copying of eight of the nine specific source code files at issue was de minimis.[180] The jury hung on whether Google's infringement of the Java API SSO constituted fair use.[181] The jury split on the special interrogatories relating to Google's equitable–estoppel defense. It found Sun/Oracle had engaged in conduct that they knew or should have known would lead Google to believe reasonably that it would not need a license to use the Java API SSO. Google nevertheless had not proven that it relied on Sun's conduct.[182]

The patent phase of the trial commenced shortly after the jury rendered its copyright verdict. The same jury ruled that Google did not infringe the eight asserted claims of the two patents at issue.[183] Therefore, the need for the third phase of the trial hinged on Judge Alsup's resolution of the post–trial copyright motions.

Shortly after the patent phase of the trial ended, Judge Alsup issued a detailed opinion holding that the Java APIs were not copyrightable,[184] resulting in dismissal of the case. Although Judge Alsup cautioned that the ruling did not hold that "Java API packages are free for all to use without license" or that "the structure, sequence, and organization of all computer programs may be stolen," the court held "on the specific facts of this case [that] the particular elements replicated by Google were free for all to use under the Copyright Act."[185]

---

179. *See* Doc. 1018, Final Charge to the Jury (Phase One) and Special Verdict Form at 12, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014) (3:10-cv-03561-WHA) (filed Apr. 30, 2012), https://docs.justia.com/cases/federal/district-courts/california/candce/3:2010cv03561/231846/1018.

180. *See* Doc. 1089, Special Verdict Form at 2, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014) (3:10-cv-03561-WHA) (filed May 7, 2012), https://docs.justia.com/cases/federal/district-courts/california/candce/3:2010cv03561/231846/1089.

181. *See id.* at 1.

182. *See id.* at 3.

183. *See* Doc. 1190, Special Verdict Form, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014) (3:10-cv-03561-WHA) (filed May 23, 2012), https://docs.justia.com/cases/federal/district-courts/california/candce/3:2010cv03561/231846/1190; Josh Lowensohn, *Jury Verdict: Android Doesn't Infringe Oracle's Patents*, CNET (May 23, 2012, 11:06 AM), http://www.cnet.com/news/jury-verdict-android-doesnt-infringe-oracles-patents/.

184. *See Oracle I*, 872 F. Supp. 2d 974 (N.D. Cal. 2012). In a pyrrhic victory for Oracle, Judge Alsup granted judgment as a matter of law holding that Google's copying of the eight test files that the jury deemed de minimis were infringing. *See* Oracle Am., Inc. v. Google Inc., No. C 10–3561, 2012 U.S. Dist. LEXIS 66417 (N.D. Cal. May 11, 2012).

185. *Oracle I*, 872 F. Supp. 2d at 1002.

Judge Alsup grounded his decision in the particular and distinctive functional attributes of the 37 Java APIs and the fact that Google wrote its own implementing code.[186] The principal copying concerned the lines of declaring code, which are necessary to operate the particular methods of the APIs. As Judge Alsup explained,

> Significantly, the rules of Java dictate the precise form of certain necessary lines of code called declarations, whose precise and necessary form explains why Android and Java *must be* identical when it comes to those particular lines of code. That is, since there is only one way to declare a given method functionality, everyone using that function must write that specific line of code in the same way.[187]

While acknowledging that the overall structure of the Java API packages is creative, original, and "resembles a taxonomy," Judge Alsup nonetheless concluded that it functions as "a command structure, a system or method of operation—a long hierarchy of over six thousand commands to carry out pre-assigned [sic] functions."[188] Applying copyright's limiting doctrines as the Ninth Circuit interprets them[189] and following CONTU's

---

186. Google did include a small (9 lines of a 3,179–line function), "innocent," and "inconsequential" segment of code (rangeCheck) in Android and eight test files that were never introduced into Android. *See Oracle I*, 872 F. Supp. 2d at 982–83. The parties stipulated, however, that there were no damages associated with these relatively modest code portions to clear the way for appeal. *See* Final Judgment, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014) (3:10-cv-03561-WHA) (filed June 20, 2012).

187. *Oracle I*, 872 F. Supp. 2d at 979 (emphasis in original). *See id.* at 981 ("In order to declare a particular *functionality*, the [Java] language *demands* that the method declaration take a particular form.") (emphasis in original); *id*. at 982 (explaining that "the names of the methods and the way in which the methods are grouped" have to be the same in order to "be interoperable. Specifically, code written for one API would not run on an API organized differently, for the name structure itself dictates the precise form of command to call up any given method.").

188. *See id.* at 999–1000.

189. Judge Alsup placed particular emphasis on *Sega Enterprises Ltd. v. Accolade, Inc.* for its rejection of the Third Circuit's broad protection for the SSO of computer software. *See* 977 F.2d 1510, 1524–25 (9th Cir. 1992) ("The *Whelan* rule . . . has been widely—and soundly—criticized as simplistic and overbroad" (citing Comput. Assocs., Inc. v. Altai, 982 F.2d 693 (2d Cir. 1992))). Judge Alsup also emphasized the *Sega* court's recognition that "the functional requirements for compatibility with [a software platform developed by another company] are not protected by copyright. 17 U.S.C. § 102(b)." *Sega*, 977 F.2d at 1522. The Ninth Circuit expressly endorsed the Second Circuit's *Altai* approach:

> Under a test that breaks down a computer program into its component subroutines and sub-subroutines and then identifies the idea or core functional element of each, such as the test recently adopted by the

guidance that when specific computer instructions, "*even though previously copyrighted, are the only and essential means of accomplishing a given task, their later use by another will not amount to an infringement*,"[190] Judge Alsup determined that Google was free to write code that accomplished the same functionality as the Java APIs at issue even if it did not achieve complete compatibility with the full Java platform.[191] In essence, Judge Alsup decided, later developers can achieve the *particular functionality* or method of operation of an API subsystem (and even groups of subsystems) so long as they write their own code and the method of writing code is not protected by a patent.

Judge Alsup's framework provided a general and concrete solution to the API copyright puzzle. Although he cautioned that his opinion was limited to the facts of the case and did not declare APIs uncopyrightable, Judge Alsup's analysis illuminated a clear pathway for software developers seeking to use APIs defined and first implemented by other software companies without running afoul of copyright law.[192] Later developers can legally use declaring code so long as they use a clean–room to implement the declarations. To many in the software industry, the ruling validated what was considered a best practice.[193] To others, it jeopardized substantial efforts and investments in developing software platforms and pioneering products, and it also threatened to undermine interoperability.[194]

> Second Circuit in *CAI*, 23 U.S.P.Q.2d at 1252–53, *many aspects of the program are not protected by copyright*. In our view, in light of the essentially utilitarian nature of computer programs, the Second Circuit's approach is an appropriate one.

Sega, 977 F.2d at 1525 (emphasis added).

190. *Oracle I*, 872 F. Supp. 2d at 986 (emphasis added by Judge Alsup) (quoting CONTU FINAL REPORT, *supra* note 21, at 20).

191. *Id.* at 1000.

192. Patent protection, trade secret law, and contractual limitations could nonetheless stand in the way, but copyright protection could not bar re–implementation of functional features of computer programs. *See* J. Jonas Anderson, *Secret Inventions*, 26 BERKELEY TECH. L.J. 917, 922 (2011); *cf.* Christian Chessman, *A "Source" of Error: Computer Code, Criminal Defendants, and the Constitution*, 105 CALIF. L. REV. 179, 210 & n.224 (2017) (noting limits to trade secret protection for software given widespread commercial reliance on open–source code).

193. *See* Wingfield & Hardy, *supra* note 2; *supra* note 69.

194. *See* Annette Hurst, *Oracle Attorney Says Google's Court Victory Might Kill the GPL*, ARS TECHNICA (May 27, 2016, 2:35 PM), http://arstechnica.com/tech-policy/2016/05/op-ed-oracle-attorney-says-googles-court-victory-might-kill-the-gpl/; Florian Mueller, *Google's 'Fair Use' Defense Against Oracle Is an Insult to Human Intelligence: Android's Use of Java APIs Violates Copyright*, FOSS PATENTS (May 22, 2016), http://www.fosspatents.com/2016/05/googles-fair-use-defense-against-oracle.html.

2. *Federal Circuit Reversal*

Notwithstanding that Oracle did not appeal any patent issue, it filed its appeal with the U.S. Court of Appeals for the Federal Circuit. The Federal Circuit has exclusive jurisdiction over appeals from district court cases involving patent infringement allegations even though, as was the circumstance in *Oracle v. Google*, neither party challenged the district court's patent rulings. The Federal Circuit is bound by regional circuit law when reviewing questions that involve law and precedent not exclusively assigned to the Federal Circuit.[195] Thus, the Federal Circuit was required to review the copyright issues according to Ninth Circuit precedents.[196]

The "software as creative expression" theme resonated with the Federal Circuit. The court's opinion highlighted the creativity of the Java APIs.[197] The court pointed to the testimony of Joshua Bloch, the former Sun software engineer whom Google referred to as its "Java guru," who "conceded" that there can be "creativity and artistry even in a single method declaration."[198] The Federal Circuit offered its own literary metaphor, noting that "the opening of Charles Dickens' *A Tale of Two Cities* is nothing but a string of short phrases. Yet no one could contend that this portion of Dickens' work is unworthy of copyright protection because it can be broken into those shorter constituent components."[199]

The Federal Circuit reversed the district court's determination that the structure, sequence, and organization of the 37 Java APIs were not copyrightable and remanded the fair use issue for retrial with revised jury

195. *See* Atari Games Corp. v. Nintendo of Am., Inc., 897 F.2d 1572, 1575 (Fed. Cir. 1990).

196. Copyright issues are not within the Federal Circuit's exclusive jurisdiction. *See* 28 U.S.C. § 1295 (2012).

197. *See Oracle II*, 750 F.3d 1339, 1352 (Fed. Cir. 2014) (explaining that the district court "acknowledged that the overall structure of Oracle's API packages is creative"); *id*. at 1356 ("The testimony at trial revealed that designing the Java API packages was a creative process and that the Sun/Oracle developers had a vast range of options for the structure and organization."); *id*. ("In its copyrightability decision, the district court specifically found that the API packages are both creative and original, and Google concedes on appeal that the originality requirements are met."); *id*. at 999 ("Yes, it is creative. Yes, it is original."); *see id.* at 1361 n.6 (noting that the Amicus Brief filed by Scott McNealy and Brian Sutphin "provide[d] a detailed example of the creative choices involved in designing a Java package"); *id*. at 1368 n.14 ("Amici McNealy and Sutphin explain that 'a quick examination of other programming environments shows that creators of other development platforms provide the same functions with wholly different creative choices.'").

198. *Oracle II*, 750 F.3d at 1339.

199. *Id.*

instructions. In reviewing the district court's determination that the Java API packages at issue were not copyrightable, the Federal Circuit distinguished between copyrightability of the declaring code and copyrightability of the structure, sequence, and organization of the API packages. The Federal Circuit ruled that the district court should not have considered the merger and *scènes à faire* doctrines when evaluating copyright subsistence because the Ninth Circuit treats these doctrines as affirmative defenses to infringement, not as limitations on copyrightability.[200] Hence, these doctrines were relevant only in determining what elements of the APIs should be filtered out in the infringement analysis.[201] Furthermore, the Federal Circuit held that the merger doctrine—which bars protection where an idea can only be expressed in one or a limited number of ways—properly focuses on the creative choices available to Sun when it created Java, not on the options available to Google when it copied Java APIs.[202] The Federal Circuit also held that the short phrases doctrine did not bar copyright protection for compilations of words and short phrases as reflected in declaring code.[203] Consequently, the appellate court ruled that copyright law protected the 7,000 lines of declaring code.

The Federal Circuit faulted the district court's reliance on *Lotus v. Borland*,[204] the First Circuit case holding that the Lotus 1–2–3 menu command hierarchy was an unprotectable "method of operation." The Federal Circuit distinguished *Lotus* on factual grounds, noting that the

200. *See id.* at 1358 (citing Ets–Hokin v. Skyy Spirits, Inc., 225 F.3d 1068, 1082 (9th Cir. 2000); Satava v. Lowry, 323 F.3d 805, 810 n.3 (9th Cir. 2003)) ("The Ninth Circuit treats *scènes à faire* as a defense to infringement rather than as a barrier to copyrightability.").

201. *See Oracle II*, 750 F.3d at 1359–62 (addressing the merger doctrine); *id*. at 1363–64 (addressing the *scènes à faire* doctrine, which Judge Alsup had rejected as a basis for holding the Java APIs to be unprotectable but that Google challenged on appeal).

202. *See id.* at 1360–61.

203. *See id.* at 1362–63. It should be noted that the district court's determination that the declaring code was uncopyrightable did not turn on the short–phrases doctrine. Judge Alsup recognized that the selection and arrangement of short phrases could be protectable. *See* Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974, 992 (N.D. Cal. 2012) (quoting Feist Publ'ns, Inc. v. Rural Tel. Serv. Co., Inc., 499 U.S. 340, 349 (1991), for the proposition that even thinly protected, factual compilations are protectable with respect to original "selection and arrangement"). Judge Alsup's ultimate determination turned on § 102(b) of the Copyright Act and interoperability. *See Oracle I*, 872 F. Supp. 2d at 997–1002.

204. Lotus Dev. Corp. v. Borland Int'l., Inc., 49 F.3d 807 (1st Cir. 1995), *aff'd without opinion by equally divided court*, 516 U.S. 233 (1996).

command labels at issue there, unlike the Java API declaring code, were "not creative" and were "essential" to operating the computer system.[205] Moreover, the Federal Circuit interpreted the Ninth Circuit's cursory opinion in *Johnson Controls, Inc. v. Phoenix Control Sys., Inc.*,[206] to hold that the SSO of a computer program is eligible for copyright protection; hence it was inconsistent with *Lotus*.[207] In so doing, the Federal Circuit resurrected the flawed analysis in the Third Circuit's *Apple* and *Whelan* cases: analyzing copyrightability of computer software based on whether the high–level function(s) of the software could be implemented in multiple ways rather than viewing a particularized set of software functions as an unprotectable "method of operation."[208]

The Federal Circuit rejected the district court's invocation of interoperability as a basis for holding the SSO of the Java APIs to be uncopyrightable. Notwithstanding the language in *Sega v. Accolade* and *Sony v. Connectix*, indicating that the precise coding to achieve interoperability is not protectable under copyright law,[209] the appellate court distinguished these cases as "focused on fair use, not copyrightability."[210] The Federal Circuit held that "copyrightability is focused on the choices available to the plaintiff at the time the computer program was created," not the defendant's desire to achieve interoperability.[211] Thus, the Federal

205. *See Oracle II*, 750 F.3d at 1365.

206. *See* Johnson Controls, Inc. v. Phx. Control Sys., 886 F.2d 1173 (9th Cir. 1989).

207. *See Oracle II*, 750 F.3d at 1365–66. The Federal Circuit's interpretation of *Johnson Controls* stretches its holding and overlooks important insights from later Ninth Circuit cases.

208. *See id.* at 1366–67.

209. *See* Sega Enters., Ltd. v. Accolade, Inc., 977 F.2d 1510 (9th Cir. 1992); Sony Comput. Entm't, Inc. v. Connectix Corp., 203 F.3d 596, 603 (9th Cir. 2000) ("There is no question that the Sony BIOS contains unprotected functional elements.").

210. *See Oracle II*, 750 F.3d at 1369 (observing that *Sega* and *Sony* never addressed whether the functional code had separable expressive elements). This assertion overlooks, however, that both courts recognized that the code that was necessary for interoperability was unprotectable and hence the copying of the entirety of the software for purposes of reverse engineering the code to determine those interoperable features constituted fair use. *See* Pamela Samuelson, *Functionality and Expression in Computer Programs: Refining the Tests for Software Copyright Infringement*, 31 BERKELEY TECH. L.J. 1215, 1278 & n.364 (2017) (noting congressional intent to facilitate reverse engineering for purposes of determining interoperability).

211. *See Oracle II*, 750 F.3d at 1370 ("[A] defendant's desire 'to achieve total compatibility . . . is a commercial and competitive objective which does not enter into the . . . issue of whether particular ideas and expressions have merged . . .'" (quoting Apple Comput., Inc. v. Franklin Comput. Corp., 714 F.2d 1240, 1253 (3d Cir. 1983))). The

Circuit explained that Google's interoperability argument comes into play only as part of a fair use defense.

The Federal Circuit leaned toward ruling in Oracle's favor on fair use, noting that "[o]n many of [Oracle's] points, Google does not debate Oracle's characterization of its conduct, nor could it on the record evidence" and that "Google knowingly and illicitly copied a creative work to further its own commercial purposes, and did so verbatim, and did so to the detriment of Oracle's market position."[212] Nonetheless, the Federal Circuit remanded the case because material facts were disputed, notably the transformativeness of the Android platform, Google's interoperability objectives, and the commercial impact of Android on Sun's/Oracle's mobile licensing activities and the potential market for a Java smartphone.[213] The Federal Circuit instructed the district court to "revisit and revise its jury instructions on fair use consistent with [the Federal Circuit's] opinion."[214]

3. The Interlocutory Certiorari Petition

Rather than seeking en banc review of the Federal Circuit's decision, Google filed a petition for a writ of certiorari with the U.S. Supreme Court.[215] Google's petition pressed the argument that the Java API declarations fall within the § 102(b) exclusion from copyright protection of methods of operation.[216] Oracle responded that the case was not appropriate for interlocutory review on substantive and prudential grounds.[217] The Supreme Court nonetheless requested the views of the Solicitor General,[218] which produced perhaps the case's most surprising filing.[219] The Solicitor General not only recommended against granting review on prudential

---

Federal Circuit follows the Third Circuit's dicta and not the apparent rejection of that position in *Sega* and *Sony*. *See Sega*, 977 F.2d at 1525; *Sony*, 203 F.3d at 603.

212. *See Oracle II*, 750 F.3d at 1376.

213. *See id*. at 1377.

214. *See id*.

215. *See* Petition for a Writ of Certiorari, Google, Inc. v. Oracle Am., Inc., 135 S. Ct. 2887 (2015), *denying cert. to* 750 F.3d 1339 (Fed. Cir. 2014) (No. 14-410), 2014 WL 5319724.

216. *See id.*

217. *See* Brief in Opposition, Google, Inc. v. Oracle Am., Inc., 135 S. Ct. 2887 (2015) (No. 14-410), 2014 WL 7205172.

218. *See* Google, Inc. v. Oracle Am., Inc., 135 S. Ct. 2887 (2015), *denying cert. to* 750 F.3d 1339 (Fed. Cir. 2014) (No. 14–410) [hereinafter *Oracle III*].

219. *See* Brief for the United States as Amicus Curiae, Google, Inc. v. Oracle Am., Inc., 135 S. Ct. 2887 (2015) (No. 14-410), 2015 WL 2457656.

grounds, but also sided with Oracle on substantive grounds.[220] The Supreme Court denied review.[221]

4. The 2016 Fair Use Retrial

The API copyright battle returned to Judge Alsup's court for a jury trial focused on applying "'the most troublesome [doctrine] in the whole law of copyright.'"[222] Google planned to assert again equitable–estoppel and laches defenses.[223] Oracle expanded the scope of its complaint to account for new Android versions, its expansion into new product areas (clothing, television, automobile, appliances, and media (Google Play)), and Android's dramatic market growth.[224]

Leading up to trial, the parties squabbled over the jury instructions on fair use.[225] After Judge Alsup adjusted the draft instructions based on input from the parties, one of the most momentous fair use jury trials in U.S. history commenced. Judge Alsup instructed the jury at the outset of the trial about the contours of the fair use doctrine, noting that the doctrine is an "equitable rule of reason" for which no generally accepted definition is possible.[226] He then read the statutory provision[227] and explained the four

220. *See id*. at 11–17; *cf*. Dan Levine & Lawrence Hurley, *Google Versus Oracle Case Exposes Differences Within Obama Administration*, REUTERS (May 15, 2015, 1:08 PM), http://www.reuters.com/article/us-google-oracle-lawsuit-insight-idUSKBN0O017Z20150515.

221. *Oracle III*, 135 S. Ct. at 2887.

222. *See Oracle II*, 750 F.3d at 1372 (quoting Monge v. Maya Magazines, Inc., 688 F.3d 1164, 1170 (9th Cir. 2012) (quoting Dellar v. Samuel Goldwyn, Inc., 104 F.2d 661, 662 (2d Cir. 1939) (per curiam))).

223. *See* Google's Trial Brief at 11–12, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012), *rev'd and remanded*, 750 F.3d 1339 (Fed. Cir. 2014), *remanded to* 118 U.S.P.Q.2d (BNA) 1561 (N.D. Cal. 2016) (3:10-cv-03561-WHA), 2016 WL 2986341 (asserting that Sun's public statements and acts approving of Android's use of Java bar enforcement of its copyrights); Order Re Willfulness and Bifurcation, Oracle Am., Inc. v. Google Inc., 131 F. Supp. 3d 946 (N.D. Cal. 2012). The equitable defenses were bifurcated and hence did not arise during the fair use trial.

224. *See* Plaintiff Oracle's [Proposed] Supplemental Complaint, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 946 (N.D. Cal. 2012), *rev'd and remanded*, 750 F.3d 1339 (Fed. Cir. 2014) (3:10-cv-03561-WHA), 2015 WL 5305164.

225. *See, e.g.*, Oracle Am., Inc., v. Google Inc., 118 U.S.P.Q.2d (BNA) 1561 (N.D. Cal. 2016) (rejecting Google's request to include "as part of a broader work" within the jury instruction defining "transformative"); Oracle's Response to the Court's Request for Critique Re Instructions on Fair Use, Oracle Am., Inc. v. Google Inc., 3:10-cv-03561-WHA (filed Apr. 14, 2016).

226. *See* Penultimate Jury Instruction on Fair Use, Oracle Am., Inc. v. Google Inc., 3:10-cv-03561-WHA (Document 1790, filed May 3, 2016).

227. *See* 17 U.S.C. § 107 (2012).

factors, boiling down the subtleties of the vast fair use jurisprudence into about a dozen treatise–like paragraphs.

The trial played out over eight grueling days of testimony ranging from the dramatic (embarrassing emails) to the mind–numbing (experts and fact witnesses explaining API design, open source, GNU, GPL, virtual machines, and distinctions between declaring and implementing code).[228] The jurors were treated to creative and strained analogies (filing cabinets, breakfast menus featuring hamburgers, and *Harry Potter* novels), all manner of demonstrative exhibits, and a witness list featuring some of Silicon Valley's most celebrated billionaires. Economic experts opined about transformativeness (from an economic, as opposed to a legal, perspective) and network effects. Both sides made witnesses squirm. The connection of some lines of questioning to copyright law's fair use factors was often tenuous. For example, Oracle devoted much of its trial time to exposing emails sent among Google engineers suggesting that they thought copyright law protected the Java APIs.

Given the large stakes—Oracle sought upwards of $10 billion in damages and injunctive relief—both sides employed top–notch trial teams and spared little expense. Building on the infringement ruling revived by the Federal Circuit, Oracle opened the second trial with a Johnnie Cochranesque rhyme: Google copied the heart of the Java platform to enter the mobile marketplace quickly and now pleads the "fair use excuse" to avoid the consequences.[229] Oracle argued that internal emails showed that Google took illegal "shortcuts" to create Android. Drawing on its successful Federal Circuit strategy, Oracle characterized the crafting of the Java API code as highly creative, and Google's copying of Java APIs as slavish and not transformative. Oracle characterized the Android team's decision to

228. Media and bloggers covered the trial closely. *See generally* Joe Mullin, ARS TECHNICA, https://arstechnica.com/author/joe-mullin/ (last visited Oct. 8, 2017); Sarah Jeong, STORIFY (Twitter username @Motherboard), https://storify.com/sarahjeong (last visited Oct. 8, 2017); FOSS PATENTS, http://www.fosspatents.com/ (last visited Oct. 8, 2017) (a blog published by Florian Mueller, a self–described "intellectual property activist"); Dante D'Orazio et al., *Google, Oracle and Java: From a Patent Spat to a Copyright Conundrum*, VERGE, https://www.theverge.com/2012/4/13/2945536/google-oracle-java-patent-copyright-lawsuit (last visited Oct. 8, 2017) (listing forty–nine articles over six years covering the case). The author also reviewed many of the exhibits that became publicly available, such as pleadings, jury instructions, and slide decks.

229. *See* Joe Mullin, *Google Took Our Property—And Our Opportunity, Oracle Tells Jury*, ARS TECHNICA (May 10, 2016), http://arstechnica.com/tech-policy/2016/05/oracle-tells-jury-dont-buy-googles-fair-use-excuse/.

forgo a license as underhanded—and breaking the WORA interoperability promise.

Google responded by emphasizing its hard work and large investment in building a transformative smartphone platform—bringing the functionality of robust web browsing, apps, and a host of other functionalities such as cameras and games (e.g., Angry Birds) to mobile devices.[230] It justified its use of Java based in part on Sun's encouragement of the developer community to use Java and its APIs. Google downplayed the expressive creativity of Java APIs by analogizing the API packages to the labels on a filing cabinet.[231] Google closed the trial by suggesting that transformativeness provides the sensible middle ground between stealing and free. "You don't have to choose between commercial and transformative . . . . [b]ecause the whole purpose of fair use is to promote innovation."

Following three days of deliberation, the jury found that Google had "shown by a preponderance of the evidence that its use in Android of the declaring lines of code and their structure, sequence, and organization from Java 2 Standard Edition Version 1.4 and Java 2 Standard Edition Version 5.0 constitutes a 'fair use' under the Copyright Act."[232] The verdict form did not ask the jury to make subsidiary factual findings.[233] With fair use decided in Google's favor, there was no need for a further damages phase. The jurors departed without comment, leaving the public and the appellate

230. *See* Joe Mullin, *Google to Jury: Android Was Built With Our Engineers' Hard Work*, ARS TECHNICA (May 10, 2016, 10:56 AM), http://arstechnica.com/tech-policy/2016/05/google-to-jury-android-was-built-with-our-engineers-hard-work/.

231. *See* Sarah Jeong, *In a $9 Billion Trial, Google's Secret Weapon Is a Filing Cabinet*, MOTHERBOARD (May 11, 2016), http://motherboard.vice.com/read/googles-lawyers-tried-to-explain-apis-to-a-jury-using-a-physical-filing-cabinet. This analogy was reminiscent of earlier API copyright cases, notably *Apple v. Microsoft* (desktop icons of the graphical user interface) and *Lotus v. Borland* (spreadsheet command labels). *See* Apple Comput., Inc. v. Microsoft Corp., 799 F. Supp. 1006 (N.D. Cal. 1992), *aff'd in part, rev'd in part*, 35 F.3d 1435 (9th Cir. 1994); Lotus Dev. Corp. v. Borland Int'l., Inc., 49 F.3d 807 (1st Cir. 1995), *aff'd without opinion by equally divided court*, 516 U.S. 233 (1996).

232. *See* Special Verdict Form, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 946 (N.D. Cal. 2012), *rev'd and remanded*, 750 F.3d 1339 (Fed. Cir. 2014) (3:10-cv-03561-WHA) (filed May 7, 2012); Joe Mullin, *Google beats Oracle—Android makes "fair use" of Java APIs*, ARS TECHNICA (May 26, 2016), http://arstechnica.com/tech-policy/2016/05/google-wins-trial-against-oracle-as-jury-finds-android-is-fair-use/.

233. *See* Special Verdict Form, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 946 (N.D. Cal. 2012), *rev'd and remanded*, 750 F.3d 1339 (Fed. Cir. 2014) (3:10-cv-03561-WHA) (filed May 7, 2012).

court without a clear understanding of how they struck the fair use balance. Judge Alsup rejected Oracle's post–trial motions seeking judgment as a matter of law and a new trial based on alleged failure to comply with discovery responsibilities. Oracle has appealed the retrial to the Federal Circuit.

III. CRITIQUE OF THE FEDERAL CIRCUIT'S 2014 COPYRIGHTABILITY DECISION

The Federal Circuit's *Oracle v. Google* decision purports to apply Ninth Circuit jurisprudence in reviewing Judge Alsup's decision holding that the compilation of functions and the structure, sequence, and organization of the Java APIs were not copyrightable. As I explore at length in a related project,[234] the Federal Circuit misinterpreted § 102(b) of the Copyright Act, misconstrued the Ninth Circuit's software copyright jurisprudence, conflated expressive and technological "creativity," and applied an overly rigid approach to copyright law's limiting doctrines. This Section summarizes the main points.

A. MISINTERPRETATION OF THE COPYRIGHT ACT

The Federal Circuit's opinion takes a broad view of the scope of copyright protection for computer software, emphasizing the low originality threshold.[235] While recognizing the § 102(b) limitations, the court did not view those constraints as applicable to copyrightability.[236] Rather, the court saw § 102(b) as only applying at the infringement and defenses stages of analysis.

The Federal Circuit misread the clear language of the Copyright Act as well as the legislative history. Section 102(b) states that "[i]n no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied in such work." A plain reading of the statute

234. *See* Menell, *supra* note 8.

235. *See Oracle II*, 750 F.3d 1339, 1354 (Fed. Cir. 2014) (explaining that the "'originality requirement is not particularly stringent.' . . . [Originality] means only that the work was independently created by the author (as opposed to copied from other works), and that it possesses at least some minimal degree of creativity.'" (quoting Feist Publ'ns, Inc. v. Rural Tel. Serv. Co., 499 U.S. 340, 345, 358 (1991))).

236. *Oracle II*, at 1354 ("[T]he district court failed to distinguish between the threshold question of what is copyrightable—which presents a low bar—and the scope of conduct that constitutes infringing activity.").

indicates that these exclusions apply at the copyrightability stage of analysis.[237] They are also pertinent to infringement analysis and the fair use defense.

Google argued that the particular compilations of functions in Java API packages were uncopyrightable "method[s] of operation." The Federal Circuit rejected the proposition that § 102(b) can be invoked in this way, quoting a comment in the legislative history of the 1976 Act stating that § 102(b) "in no way enlarges or contracts the scope of copyright protection," but merely "restates . . . that the basic dichotomy between expression and idea remains unchanged."[238]

That dichotomy traces back to the Supreme Court's seminal decision in *Baker v. Selden*,[239] which held that the owner of copyright in a book disclosing a method of accounting could not bar others from using the methods disclosed in the book unless they had patent protection.[240] The Supreme Court did not inquire into whether there were other methods that achieved the same general purpose (bookkeeping). Rather, the Court categorically excluded any claim to a method of accounting even as it ruled that Selden's accounting book describing the method was copyrightable.[241] The CONTU Report concurs: "one is always free to make a machine

237. Courts routinely apply the analogous separability analysis of the useful article doctrine at the copyrightability stage. *See* Menell, *supra* note 8.

238. *Oracle II*, 750 F.3d at 1356 (quoting Feist Publ'ns, Inc. v. Rural Tel. Serv. Co., 499 U.S. 340, 356 (1991)).

239. Baker v. Selden, 101 U.S. 99 (1879).

240. *See id*. at 102 ("To give to the author of the book an exclusive property in the art described therein, when no examination of its novelty has ever been officially made, would be a surprise and a fraud upon the public. That is the province of letters-patent, not of copyright. The claim to an invention or discovery of an art or manufacture must be subjected to the examination of the Patent Office before an exclusive right therein can be obtained; and it can only be secured by a patent from the government.").

241. The Federal Circuit attempts to fit *Baker v. Selden* into its atextual reading of § 102 by stating that, "The [Supreme] Court [in *Baker v. Selden*] indicated that, if it is necessary to use the forms Selden included in his books to make use of the accounting system, that use would not amount to copyright infringement. *See* [*Baker v. Selden*, 101 U.S. at 104] (noting that the public has the right to use the account-books and that, 'in using the art, the ruled lines and headings of accounts must necessarily be used as incident to it')." *Oracle II*, 750 F.3d at 1355. A faithful reading of *Baker v. Selden* recognizes that the Court held that the accounting method was uncopyrightable, not merely not infringed. *See* Nicholas M. Lampros, *Leveling Pains: Clone Gaming and the Changing Dynamics of an Industry*, 744 BERKELEY TECH. L.J. 743, 755 (2013); Lemley, *supra* note 46, at 5. That is the essence of the idea–expression dichotomy. *See id.*

perform *any* conceivable process (in the absence of a patent)" so long as one does not "take another's program."[242]

In accordance with this principle, Google was entitled to make a mobile device ("a machine") perform the same functions as a Java API package (a "conceivable process") with clean–roomed computer code (not "another's program"). Each Java API package constituted a particular subsystem within a larger particular computing environment. Hence, Google was justified in selecting a set of Java API packages and implementing them with original code to create a new machine.

B. MISREADING NINTH CIRCUIT JURISPRUDENCE

Beyond misconstruing § 102(b), the Federal Circuit's opinion diverges from the clear language and evolution of the Ninth Circuit's software copyright jurisprudence. Judge Alsup drew principally from the First Circuit's *Lotus* decision and the Ninth Circuit's *Sega* decision in framing his analysis. The Federal Circuit held that the *Lotus* decision is "inconsistent" with Ninth Circuit precedent[243] and that the *Sega* decision is inapt.[244] Neither of these interpretations, however, withstands scrutiny.

Although the Ninth Circuit has not had occasion to address the *Lotus* line of analysis specifically, it holds that software code that is necessary for interoperability is not copyrightable. In *Sega v. Accolade*, the Ninth Circuit stated, "the functional requirements for compatibility with the Genesis [platform are] aspects of Sega's programs that are not protected by copyright. 17 U.S.C. § 102(b)."[245] Such aspects of the Genesis video game platform are functional specifications of the computer system—a relatively simple API. The Ninth Circuit unequivocally ruled that the interface specification was not copyrightable, which parallels the *Lotus* analysis. The Ninth Circuit could not have cited the First Circuit's *Lotus* decision because that decision was not handed down until several years later.

Not only did the Federal Circuit misread the Ninth Circuit's *Sega* and *Sony* decisions, it embraced analyses that the Ninth Circuit has expressly rejected. By holding that the code for interoperability may be protectable, the Federal Circuit resurrects the Third Circuit's dicta in *Apple v. Franklin*: "courts have recognized that, once the plaintiff creates a copyrightable work, a defendant's desire 'to achieve total compatibility . . . is a

242. *See* CONTU FINAL REPORT, *supra* note 21, at 20 (emphasis added).
243. *Oracle II*, 750 F.3d at 1365.
244. *Id.* at 1369.
245. Sega Enters. v. Accolade, Inc., 977 F.2d 1510, 1522 (9th Cir. 1992).

commercial and competitive objective which does not enter into the . . . issue of whether particular ideas and expressions have merged.'"[246] To the contrary, the Ninth Circuit holds that copyright law does not stand in the way of achieving interoperability. As noted earlier,[247] the Third Circuit comment is dicta as Franklin Computer had copied the entirety of Apple's computer programs. More importantly, § 102(b), the CONTU Report, and the *Sega*/*Sony* decisions directly contradict the Third Circuit's proposition.

Moreover, the Federal Circuit endorsed and followed the Third Circuit's *Apple/Whelan* framework, holding that everything not necessary to the *general* purpose or function of a work is protectable expression: "We agree with Oracle that, under Ninth Circuit law, an original work—even one that serves a function—is entitled to copyright protection as long as the author had multiple ways to express the underlying idea."[248] The Federal Circuit credited Oracle's statement that it only claimed "its *particular* way of naming and organizing each of the 37 Java API packages" and that it "'cannot copyright the idea of programs that open an internet connection,' but 'it can copyright the precise strings of code used to do so, at least so long as "other language is available" to achieve the same function.'"[249] In an accompanying footnote, the court noted that Oracle's counsel explained at oral argument that Oracle "would never claim that anyone who uses a package-class-method manner of classifying violates our copyright. We don't own every conceivable way of organizing, we own only our specific expression—our specific way of naming each of these 362 methods, putting them into 36 classes, and 20 subclasses."[250] The Federal Circuit reasoned that as long as the same general functions could be accomplished using different code, then the first author's code for such general functions was protectable.[251]

---

246. *See Oracle II*, 750 F.3d at 1357 (quoting Apple Comput. v. Franklin Comput. Corp., 714 F.2d 1240, 1253 (3d Cir. 1983)).

247. *See supra* note 28 and accompanying text.

248. *See Oracle II*, 750 F.3d at 1371 (quoting Apple Comput., Inc. v. Franklin Computer Corp., 714 F.2d 1240, 1253 (3d Cir. 1983)); *see also id.* at 1366 (noting that the Third Circuit in *Apple v. Franklin* "focused 'on whether the idea is capable of various modes of expression' and indicated that, '[i]f other programs can be written or created which perform the same function as [i]n Apple's operating system program, then that program is an expression of the idea and hence copyrightable'" (quoting Apple Comput., Inc. v. Franklin Computer Corp., 714 F.2d 1240, 1252 (3d Cir. 1983)).

249. *See Oracle II*, 750 F.3d at 1367–68 (quoting from Oracle's Reply Brief).

250. *Id.* at 1368 n.13.

251. *See id.* at 1356 (setting the foundation for its analysis by observing that "the Sun/Oracle developers had a vast range of options for the structure and organization" of

While this mode of analysis comports with Ninth Circuit jurisprudence with regard to implementing code, it contradicts copyright law principles and Ninth Circuit precedent as regards declarations that are necessary to operate a particular computing system. Contrary to the Third Circuit's dicta in *Apple v. Franklin*, the Ninth Circuit's *Sega* and *Sony* decisions hold that the code necessary for interoperability is uncopyrightable.[252] Thus, a defendant's desire to achieve compatibility does enter into the issue of whether particular ideas and expressions have merged in the Ninth Circuit. It resolves the issue so long as the defendant independently writes the code to achieve the particular functions of the plaintiff's software. Secondly, the *Sega* decision unequivocally rejects the *Whelan* framework of simply asking whether there are multiple ways of programming a particular function: "[t]he *Whelan* rule . . . has been widely—and soundly—criticized as simplistic and overbroad."[253] The *Sega* court instead recognized that "the functional requirements for compatibility with [a software platform developed by another company] are not protected by copyright."[254]

---

the Java APIs); *id*. at 1360 ("We have recognized . . . applying Ninth Circuit law, that the 'unique arrangement of computer program expression . . . does not merge with the process so long as alternate expressions are available.'" (quoting Atari Games Corp. v. Nintendo of Am., Inc., 975 F.2d 832, 840 (Fed. Cir. 1992)); *id*. ("Because Nintendo produced expert testimony 'showing a multitude of different ways to generate a data stream which unlocks the NES console,' we concluded that Nintendo's specific choice of code did not merge with the process."); *id*. at 1360 n.5 ("It is undisputed that Microsoft and Apple developed mobile operating systems from scratch, using their own array of software packages."); *id*. at 1368 n.14 (referencing the amicus brief of former Sun executives explaining that "a quick examination of other programming environments [Apple's iOS and Microsoft's Windows Phone] shows that creators of other development platforms provide the same functions with wholly different creative choices").

252. *See* Sega Enters. v. Accolade, Inc., 977 F.2d 1510, 1522 (9th Cir. 1992) (holding that "the functional requirements for compatibility with [a software platform developed by another company] are not protected by copyright" under 17 U.S.C. § 102(b)).

253. *See id.* at 1525 (citing Comput. Assocs. Inc. v. Altai, 23 U.S.P.Q.2d at 1252 (2d Cir. 1992), *withdrawn and superseded*, 982 F.2d 693 (2d Cir. 1992)).

254. *See id.* at 1522 (citing 17 U.S.C. § 102(b) (2012)).

C. CONFLATION OF EXPRESSIVE AND TECHNOLOGICAL "CREATIVITY"

The Federal Circuit embraced Oracle's argument (and that of former Sun executives)[255] that API design is a "creative," "noble craft"[256] entitled to robust protection. Oracle analogized API design to the drafting of *Harry Potter* novels.[257]

This argument, however, conflates idea and expression. APIs function as the levers and gears of digital machines. The declarations must be reproduced to replicate the particular functionality. Android programmers needed to reproduce the same package, class, and method names to allow their programs to respond to the same inputs and to produce the same outputs as Java API packages.

Protection for a particular combination of functions effectively monopolizes that technological solution. The digital revolution has taught us that once consumers and programmers become accustomed to a particular interface specification, robust intellectual property protection for APIs can have dramatic effects on competition and innovation.[258] The Supreme Court and Congress have determined that inventors must meet the patent law's higher thresholds of novelty, nonobviousness, and disclosure to garner protection for technological creativity. Furthermore, patent protection is limited to twenty years from the filing of the application, far less than copyright law's ninety–five year duration for corporate authors.

D. OVERLY RIGID APPROACH TO LIMITING DOCTRINES

The Federal Circuit erred by attempting to shoehorn analysis of API design into a framework designed for analyzing software code. As the *Lotus* court and Judge Alsup recognized, copyright law does not dictate a monolithic approach to all media. Copyright law has long relied on a

255. *See* Corrected Brief of Scott McNealy and Brian Sutphin as Amici Curiae in Support of Reversal, Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339 (Fed. Cir. 2014), *rev'g* 872 F. Supp. 2d 974 (N.D. Cal. 2012) (3:10-cv-03561-WHA), 2013 WL 942809 (characterizing API design as a highly creative process in which programmers work from a limitless pallet of choices).

256. *See* Opening Brief and Addendum of Plaintiff-Appellant at 12–13, 72, Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339 (Fed. Cir. 2014), *rev'g* 872 F. Supp. 2d 974 (N.D. Cal. 2012) (3:10-cv-03561-WHA), 2013 WL 518611.

257. *See id.*; *Oracle II*, 750 F.3d at 1356 (citing the district court's copyrightability decision for the proposition that "[t]he overall name tree of the Java API, of course, has creative elements").

258. *See generally* CARL SHAPIRO & HAL R. VARIAN, INFORMATION RULES: A STRATEGIC GUIDE TO THE NETWORK ECONOMY 103–226 (1999); Menell, *supra* note 12.

common law approach for adapting the law to deal with new technologies and other dynamic considerations.[259] The idea–expression dichotomy provides flexibility in the domain of functional works. Courts need to be sensitive to technological nuance in applying § 102(b), and to take care in evolving the family of doctrines (merger, *scènes à faire*, idea–expression dichotomy, *Baker v. Selden*, and fair use) on which it is based.

Oracle v. Google is the first litigated copyright case since *Lotus* to focus specifically on copyright protection for API design.[260] Judge Alsup saw that although the Ninth Circuit had endorsed the *Altai* framework for cases involving implementing code, the *Oracle v. Google* case required an alternative framework to address API design. He recognized that *Lotus* provided pertinent analysis and that *Sega* addressed the uncopyrightability of code necessary for interoperability. His decision thoughtfully combined these elements to produce a sound framework.[261]

The scope of protection for computer software brought new issues to the fore. When *Sega* developed its lockout code for the Genesis game console, there were no constraints on the arbitrary string characters that it designated for the key. Just as bank customers can choose whatever PIN they like (within the field constraint of four numbers), Sega was free to choose an arbitrary string of letters, numbers, and symbols to lock and unlock its platform. The Ninth Circuit nevertheless determined that § 102(b) did not protect the lockout code because once it was "created" for use as lockout code, it became functional.

---

259. *See* Menell, *supra* note 26, at 70.

260. As noted above, the *Sega* case addressed this issue as part of a fair use analysis of intermediate copying of software code. *See supra* notes 47–52. This API design issue has, however, arisen in litigation contexts, but was not resolved by judicial decisions. As noted earlier, *see supra* note 96, Sun Microsystems sued Microsoft for breach of contract, copyright infringement, and trademark infringement relating to Microsoft's efforts to fragment the Java platform in the late 1990s. Express Logic sued Green Hills for alleged copyright infringement of the API for real–time operating system software. *See Express Logic Seeks Injunction Against Green Hills*, EE TIMES (June 12, 2006, 6:00 PM), http://www.eetimes.com/document.asp?doc_id=1161911. The arbitration panel interpreted Ninth Circuit law very similarly to Judge Alsup and found the declaring code (header files) at issue in that case to be uncopyrightable. *See* Patrick Mannion, *Ruling for Green Hills Clears Way for Copying of APIs*, EE TIMES (Aug. 21, 2007, 9:00 PM), http://www.eetimes.com/document.asp?doc_id=1166905 (reporting that the arbitration panel held that copyright laws do not extend to the functionality of APIs in a dispute involving real time operating systems). As a disclaimer, the author served as a consultant for Sun Microsystems in Sun's litigation against Microsoft and as an expert witness for Green Hills in the litigation brought by Express Logic. The author was compensated by the parties that retained him in these matters.)

261. *See Oracle I*, 872 F. Supp. 2d 974, 984–97 (N.D. Cal. 2012).

The First Circuit reached a similar conclusion in the *Lotus* case. At the time that Lotus designed its menu command hierarchy for the Lotus 1–2–3 program, there were numerous options for labeling the functions and countless compilations of function names. Once programmers of macros for the Lotus spreadsheet became accustomed to those function names, however, the labels took on tremendous importance to users. To bestow copyright protection on such a system would potentially confer outsized market power over the particular method of operating a spreadsheet due to users' high switching costs—many had developed sophisticated macros for automating their accounting and other record keeping. The First Circuit recognized that this issue was best addressed at the copyrightability stage. Like Selden's accounting book, Lotus's spreadsheet program was entitled to copyright protection at the moment it was created (or in the case of Selden's book, when the applicable formalities at the time were met) but the method of operation (like Selden's accounting system) remained outside of copyright protection.

Although more sophisticated than an ATM PIN, the Genesis lockout code, or even Lotus's menu–command hierarchy, the declarations of the Java APIs similarly functioned as methods of operating particular digital machines—packages of functions. Judge Alsup's focus on § 102(b) and the *Lotus* court's framework better address the copyright issues in *Oracle v. Google* than the *Altai* framework, which was developed for analyzing copyright code.

By rigidly focusing on Ninth Circuit cases that treat the merger and *scènes à faire* doctrines as defenses to infringement rather than copyrightability doctrines,[262] the Federal Circuit missed the forest for the trees. Section 102(b) can operate as both a threshold doctrine and as part of the filtration step of infringement analysis. In fact, in the *Ets–Hokin* case, on which the Federal Circuit bases its analysis, the Ninth Circuit treats the bottle that is the object of the photograph in question as uncopyrightable under the useful–article doctrine, i.e., at a threshold copyrightability

262. The Federal Circuit cites *Ets–Hokin v. Skyy Spirits, Inc.*, 225 F.3d 1068, 1073, 1082 (9th Cir. 2000) (involving photography), and *Satava v. Lowry*, 323 F.3d 805, 810 n.3 (9th Cir. 2003) (involving glass–encased jellyfish sculptures and holding that "[t]he Ninth Circuit treats *scènes à faire* as a defense to infringement rather than as a barrier to copyrightability").

level.[263] Copyright law, like the patent law's nonobviousness doctrine, does not fit a rigid mold.[264]

E. TREATING API DESIGN AS VARIABLE EXPRESSION RATHER THAN UNIQUE FUNCTION

The Federal Circuit erred in treating the set of 37 Java API declarations as "source code" rather than as the functional specifications for a particular computer system.[265] Such API design defines the particular data–processing capabilities of a particular computing machine and is necessary for another virtual machine to perform the same processes.

From a copyright standpoint, the critical question is whether a particular set of instructions, expressed in a particular way, is "the only and essential means of accomplishing a given task."[266] Alternatively, are these particular instructions, expressed in this particular way, the only way to effectuate "the actual processes or methods embodied in the program"?[267] As CONTU explained, "one is always free to make a machine perform *any* conceivable process (in the absence of a patent)" so long as one does not "take another's program."[268] The test is not whether there are multiple ways of writing code to perform a *general* purpose. Congress instead viewed the idea–expression dichotomy as enabling anyone to build a machine capable of performing any *particular* function, including those for which others had written computer code. Under the idea–expression dichotomy, copyright protection must not lock competitors out of a particular platform—only patent protection can. Copyright protection can only require that competitors write their own implementing code. If the only way to achieve such a "certain result"[269] includes literally copying even detailed textual–represented

263. *See Ets–Hokin*, 225 F.3d at 1073, 1080.

264. *Cf.* KSR Int'l Co. v. Teleflex, Inc., 550 U.S. 398 (2007) (reversing the Federal Circuit for applying too rigid a test—the teaching–suggestion–motivation requirement—for analyzing patent law's nonobviousness doctrine).

265. *See Oracle II*, 750 F.3d 1339, 1368 (Fed. Cir. 2014) ("Given the [district] court's findings that the SSO is original and creative, and that the declaring code could have been written and organized in any number of ways and still have achieved the same functions, we conclude that Section 102(b) does not bar the packages from copyright protection just because they also perform functions.").

266. *See* CONTU FINAL REPORT, *supra* note 21, at 20.

267. *See* H.R. REP. No. 94-1476, at 57 (1976).

268. *See* CONTU FINAL REPORT, *supra* note 21, at 20 (emphasis added).

269. As added in the 1980 amendments, the Copyright Act defines a "computer program" as "set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result." 17 U.S.C. § 101 (2012).

information, such as declarations, then copyright law does not stand in the way.

Google followed this path. It sought to achieve the particular functionalities of 37 Java API packages. After negotiations to license the Java APIs reached an impasse, Google independently wrote its own implementing code. Oracle does not dispute that Google needed to include the particular declarations to make its Android platform perform the particular functions of the 37 Java APIs. Thus, the Federal Circuit should have affirmed Judge Alsup's copyrightability ruling, and the case should have ended at that stage.

IV. THE *ORACLE V. GOOGLE* LITIGATION: FUTURE PATHWAYS

The *Oracle v. Google* fair use jury trial ranks among the most significant computer software intellectual property trials and copyright fair use trials in U.S. history. Yet, it did little to clarify intellectual property protection for computer software. Even though Google has prevailed thus far, the jury's fair use decision has little precedential significance. The jury's verdict in *Oracle v. Google* does not insulate other technology companies from the risk of copyright liability for independently implementing the code necessary to achieve particular functionality. Nor does it stand in the way of Oracle filing a new complaint alleging that new versions of Android infringe copyright protection for the structure of Java API packages.

The only secure safe harbors are to develop an independent platform or license the pre–existing APIs, each of which can have undesirable effects. Independent platforms raise compatibility and interoperability concerns, risk fragmentation of markets, and reduce positive externalities from network effects. The need to negotiate a license erects a barrier to entry and risks market exclusion and vertical monopolization.

Thus, notwithstanding six years of litigation and two trials, the *Oracle v. Google* litigation has contributed to, rather than quelled, confusion surrounding API copyright protection. Fair use is a highly unpredictable doctrine. Legal advisors will need to inform clients that there is no clear safe harbor for reimplementing APIs short of a license. Other trial teams will face the same troublesome doctrines when confronted with other sets of complex facts.

Furthermore, by resolving the fair use question by a simple jury verdict form,[270] the *Oracle v. Google* litigation sheds little light on the reasoning on which the jury based its decision. There were no formal factual findings. Therefore, the decision contributes little to our understanding of the fair use factors—transformativeness, commerciality, nature of the copyrighted work—or how they are balanced in the context of APIs. All we know is that Google's particular reimplementation for particular devices was fair use. As the motion for a new trial reveals,[271] however, new versions of the Android platform could provide the basis for a new copyright infringement action.

Such uncertainty can be especially problematic for technology companies. The design of a new platform requires planning. Network economics teaches that the viability and value of a platform depends critically upon its ability to leverage consumers' and programmers' familiarity with APIs.[272] Yet the current status of API copyright jurisprudence hinges liability for copyright infringement on fair use—"the most troublesome [doctrine] in the whole law of copyright."[273] And as the *Oracle v. Google* litigation has already illustrated, a jury verdict does not necessarily resolve a dispute. This is especially true in a case in which the cost of appeal is relatively low in comparison with the stakes involved and where the parties do not perceive advantages in a settlement.[274]

As Google completed its case in chief, Oracle filed a motion requesting that Judge Alsup render judgment as a matter law ("JMOL"). Judge Alsup

---

270. *See* Notice of Final Charge to the Jury (Phase One) and Special Verdict Form, Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012), *rev'd and remanded*, 750 F.3d 1339 (Fed. Cir. 2014), *remanded to* 118 U.S.P.Q.2d (BNA) 1561 (N.D. Cal. 2016) (3:10-cv-03561-WHA).

271. *See* Oracle's Rule 59 Motion for a New Trial, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012), *rev'd and remanded*, 750 F.3d 1339 (Fed. Cir. 2014), *remanded to* 118 U.S.P.Q.2d (BNA) 1561 (N.D. Cal. 2016) (3:10-cv-03561-WHA), 2016 WL 9045812.

272. *See* Menell, *supra* note 8.

273. *See Oracle II*, 750 F.3d at 1372 (quoting Monge v. Maya Magazines, Inc., 688 F.3d 1164, 1170 (9th Cir. 2012) (quoting Dellar v. Samuel Goldwyn, Inc., 104 F.2d 661, 662 (2d Cir. 1939) (per curiam))); *see also* PAUL GOLDSTEIN, GOLDSTEIN ON COPYRIGHT § 12.1 (3d ed. 2005) ("No copyright doctrine is less determinate than fair use."); David Nimmer, *"Fairest of Them All" and Other Fairy Tales of Fair Use*, 66 LAW & CONTEMP. PROBS. 263, 263 (2003).

274. *See supra* notes 154–163 and accompanying text; Dan Levine, *Oracle Suit Versus Google at Settlement Impasse: Judge*, CHI. TRIB. (Apr. 2, 2012), http://articles.chicagotribune.com/2012-04-02/business/sns-rt-us-oracle-google-lawsuitbre8310zk-20120402_1_oracle-s-java-oracle-suit-google.

rejected Oracle's JMOL motion.[275] The court erred on Oracle's side in allowing an instruction on the propriety of the defendant's conduct,[276] notwithstanding that the Federal Circuit did not call attention to this consideration in its remand decision and the Supreme Court's decision in *Campbell v. Acuff–Rose Music, Inc.* downplays or jettisons this consideration.[277] Judge Alsup explained that, based on the evidence presented, the jury could well have determined that it was fair use to maintain the same structure of 37 Java API packages in the Android reimplemented packages so as to avoid the confusion that would ensue from scrambling the various functions: "avoiding cross-system babel promoted the progress of science and useful arts—or so our jury could reasonably have found."[278]

Judge Alsup rejected Oracle's arguments that Android's use of the Java APIs should have been deemed "entirely commercial" and nontransformative and that the Java APIs should have been considered "highly creative" because of the myriad ways in which the functions could have been implemented. With respect to the fourth fair use factor—the impact on the potential market for the Java platform—Judge Alsup ruled that the jury "could reasonably have found that use of the declaring lines of code (including their SSO) in Android caused no harm to the market for the copyrighted works, which were for desktop and laptop computers" and that

275. *See* Order Denying Rule 50 Motions, Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012), *rev'd and remanded*, 750 F.3d 1339 (Fed. Cir. 2014), *remanded to* 118 U.S.P.Q.2d (BNA) 1561 (N.D. Cal. 2016) (3:10-cv-03561-WHA), 2016 WL 3181206.

276. *See* Notice of Final Charge to the Jury (Phase One) and Special Verdict Form at § 27, Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012), *rev'd and remanded*, 750 F.3d 1339 (Fed. Cir. 2014), *remanded to* 118 U.S.P.Q.2d (BNA) 1561 (N.D. Cal. 2016) (3:10-cv-03561-WHA) (filed May 20, 2016).

277. *See* Campbell v. Acuff-Rose Music, Inc., 510 U.S. 569, 585 n.18 (1994) ("Even if good faith were central to fair use, 2 Live Crew's actions do not necessarily suggest that they believed their version was not fair use; the offer [to license the plaintiff's work] may simply have been made in a good-faith effort to avoid this litigation. If the use is otherwise fair, then no permission need be sought or granted."); PAUL GOLDSTEIN, 2 GOLDSTEIN ON COPYRIGHT § 12.2.2, at 12:44.5–12:45 (3d ed. 2005 & Supp. 2016).

278. Order Denying Rule 50 Motions at 8–10, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012), *rev'd and remanded*, 750 F.3d 1339 (Fed. Cir. 2014), *remanded to* 118 U.S.P.Q.2d (BNA) 1561 (N.D. Cal. 2016) (3:10-cv-03561-WHA), 2016 WL 3181206. Judge Alsup further explained that intersystem consistency "differs from the interoperability point criticized by the Federal Circuit. The immediate point of cross-system consistency focuses on avoiding confusion in usage between the two systems, both of which are Java-based, not on one program written for one system being operable on the other, the point addressed by the Federal Circuit." *Id.* at 10 n.6.

the copying had little effect on licensing of Java ME beyond "the tailspin already predicted within Sun."[279] The court concluded its ruling by highlighting the contradiction between Oracle's pretrial instruction arguments—focusing on characterizing the fair use test as an equitable rule of reason affording juries broad discretion based on the contextual facts of the case—and its JMOL motion urging that the court override the jury's balancing of the fact–specific factors:

> In applying an "equitable rule of reason," our jury could reasonably have given weight to the fact that cross-system confusion would have resulted had Google scrambled the SSO and specifications. Java programmers and science and the useful arts were better served by a common set of command-type statements, just as all typists are better served by a common QWERTY keyboard.[280]

That decision did not, however, end even the trial court phase of the litigation. Oracle filed a new JMOL motion in early July that critiqued Judge Alsup's rejection of its first JMOL motion.[281] More significantly, Oracle filed a motion requesting a new trial based on Google's alleged failure to disclose its plan to install Android Marshmallow on desktop and laptop computers.[282] In its reply to Google's opposition,[283] Oracle contended that the withheld evidence "directly refutes Google's argument to the jury that 'Android is not a substitute [because] Java SE is on personal computers; Android is on smartphones.'"[284] Judge Alsup rejected these motions but left open the option for Oracle to file a new copyright infringement complaint

279. *See id*. at 17.

280. *See id*. at 18.

281. *See* Oracle's Rule 50(b) Motion for Judgment as a Matter of Law, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012), *rev'd and remanded*, 750 F.3d 1339 (Fed. Cir. 2014), *remanded to* 118 U.S.P.Q.2d (BNA) 1561 (N.D. Cal. 2016) (3:10-cv-03561-WHA), 2016 WL 9045685.

282. *See* Oracle's Rule 59 Motion for a New Trial, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012), *rev'd and remanded*, 750 F.3d 1339 (Fed. Cir. 2014), *remanded to* 118 U.S.P.Q.2d (BNA) 1561 (N.D. Cal. 2016) (3:10-cv-03561-WHA), 2016 WL 9045812.

283. *See* Google Inc.'s Opposition to Oracle's Rule 59 Motion for a New Trial, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012), *rev'd and remanded*, 750 F.3d 1339 (Fed. Cir. 2014), *remanded to* 118 U.S.P.Q.2d (BNA) 1561 (N.D. Cal. 2016) (3:10-cv-03561-WHA), 2016 WL 9045809.

284. *See* Oracle's Reply in Support of its Rule 59 Motion for a New Trial at 1, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012), *rev'd and remanded*, 750 F.3d 1339 (Fed. Cir. 2014), *remanded to* 118 U.S.P.Q.2d (BNA) 1561 (N.D. Cal. 2016) (3:10-cv-03561-WHA), 2016 WL 9045811.

based on Google's implementations of Android in devices other than smartphones and tablets.[285] Oracle has appealed the fair use decision.[286]

Oracle has reason for optimism about a Federal Circuit appeal.[287] Under the Federal Circuit's Internal Operating Procedures, the same panel that reversed Judge Alsup's copyrightability ruling and set forth guiding principles for the fair use trial will likely hear the appeal of the fair use trial.[288] Oracle has preserved various objections to Judge Alsup's jury instructions. Oracle can also pursue the district court's denial of its new trial motion. Should Oracle prevail, it will have the opportunity to learn from what will have become an expensive mock trial. It can potentially improve some of its themes and better prepare its witnesses. Moreover, Google will be prevented from asserting one of its key arguments—that Android is not a substitute for Java SE on personal computers. Alternatively, Google might decide to redesign its Chrome integration with Android to work around the 37 Java APIs. But even if Google does so, its fair use argument may be weakened, especially if it integrates its mobile and desktop platforms. The appellate panel has already indicated that there was much force to Oracle's position and that many of the facts relevant to fair use were not in dispute.[289]

---

285. *See* Order Denying Renewed Motion for Judgment as a Matter of Law and Motion for a New Trial at 5, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012), *rev'd and remanded*, 750 F.3d 1339 (Fed. Cir. 2014), *remanded to* 118 U.S.P.Q.2d (BNA) 1561 (N.D. Cal. 2016) (3:10-cv-03561-WHA), 2016 WL 5393938.

286. *See* Joe Mullin, *It's Official: Oracle Will Appeal Its "Fair Use" Loss Against Google*, ARS TECHNICA (Oct. 27, 2016), http://arstechnica.com/tech-policy/2016/10/its-official-oracle-will-appeal-its-fair-use-loss-against-google/.

287. *See* Florian Mueller, Oracle v. Google: *Jury Finds in Favor of "Fair Use," as No Reasonable, Properly-Instructed Jury Could Have*, FOSS PATENTS (May 26, 2016), http://www.fosspatents.com/2016/05/oracle-v-google-jury-finds-in-favor-of.html (contending that Judge Alsup's instructions set the fair use bar far too low). *But see* Jonathan Band, *Sanity Prevails Again, Part II: The District Court Leaves the* Oracle v. Google *Fair Use Verdict in Place*, DISRUPTIVE COMPETITION PROJECT (June 10, 2016), http://www.project-disco.org/intellectual-property/061016-sanity-prevails-again-part-ii-the-district-court-leaves-the-oracle-v-google-fair-use-verdict-in-place (contending that "given how the district court meticulously found evidence in the record supporting the reasonableness of the jury's fair use finding, it is hard to imagine that the Federal Circuit will reverse it").

288. *See* U.S. CT. OF APPEALS FOR THE FED. CIRCUIT, INTERNAL OPERATING PROCEDURES 11–12 (2008), http://www.cafc.uscourts.gov/sites/default/files/IOPs122006.pdf ("When an appeal is docketed in a case that was previously remanded by this court . . . the clerk's office attempts to assign the appeal to the previous panel, to a panel including at least two members of the previous panel (if one of those members was the authoring judge), or to a panel that contains the authoring judge, if such a panel is otherwise constituted and available on a subsequent argument calendar.").

289. *See Oracle II*, 750 F.3d at 1376.

Google also has reason for optimism. First, it won the jury trial after Judge Alsup modified the jury instructions in light of the parties' concerns. Second, even if Google again lost at the Federal Circuit, it could petition the Supreme Court to review the Federal Circuit's API copyrightability ruling,[290] which could expand Android's reach and remove the cloud of future infringement lawsuits.

Assuming that the parties cannot reach a settlement, the Federal Circuit will review the fair use trial and post–trial rulings. Should Google prevail, Oracle would likely take a shot at Supreme Court review. Google would have the option of reasserting the copyrightability issue. Alternatively, the Federal Circuit could remand for another fair use trial or resolve the ultimate fair use question in Oracle's favor, thereby setting up a Google writ of certiorari petition raising both API–copyrightability and fair–use questions. Thus, even in the most optimistic scenario, the case will drone on for several more years.

V. DEBUGGING APPELLATE INTELLECTUAL PROPERTY JURISDICTION

The unusual jurisdictional posture of the *Oracle v. Google* case highlights an overlooked defect of appellate intellectual property jurisdiction. When Congress established the Court of Appeals for the Federal Circuit in 1982, it sought to address confusion in patent jurisprudence and the forum shopping that it generated.[291] Legislators did not, however, provide a procedure for reviewing Federal Circuit interpretations of regional circuit law short of Supreme Court review. Forum shopping motivated by conflicting regional circuit patent jurisprudence dominated the policy discussion. By contrast, computer software litigation was in its infancy, and the patentability of computer software was in flux.[292] Thus, it is not surprising that Congress did not put

290. *See* Google's Trial Brief at 8 n.12, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012), *rev'd and remanded*, 750 F.3d 1339 (Fed. Cir. 2014), *remanded to* 118 U.S.P.Q.2d (BNA) 1561 (N.D. Cal. 2016) (3:10-cv-03561-WHA), 2016 WL 2986341 ("Google does not waive and hereby expressly preserves its position that the SSO/declarations are not protected by copyright law. *See, e.g.*, Bikram's Yoga Coll. of India, L.P. v. Evolation Yoga, LLC, 803 F.3d 1032 (9th Cir. 2015).").

291. *See infra* notes 294–304 and accompanying text.

292. *Compare* Diamond v. Diehr, 450 U.S. 175 (1981) (viewing software claims as a whole as eligible for patent protection so long as there is post–solution activity), *with* Parker v. Flook, 437 U.S. 584 (1978) (holding that a claim to a computer program is

in place the types of checks and balances that might be needed to avoid or limit jurisprudential confusion in non–patent aspects of the Federal Circuit's jurisdiction.

With the emergence of both software patenting and copyright protection for computer software, it was only a matter of time before Federal Circuit and regional circuit copyright jurisprudence would intersect. The *Oracle v. Google* case illustrates the "forking"[293] of Ninth Circuit copyright jurisprudence. Whereas Judge Alsup placed principal reliance on the Ninth Circuit's *Sega* decision, which expressly rejected the *Whelan* framework, the Federal Circuit emphasized the *Nintendo v. Atari Games* decision,[294] a prior Federal Circuit decision applying Ninth Circuit law. That decision predates *Sega* and builds on the inchoate foundation of the Ninth Circuit's *Johnson Controls* decision.

The *Oracle v. Google* litigation reveals the jurisprudential confusion that can arise from surrogate interpretation of judicial decisions. En banc review provides a mechanism for addressing intra–circuit splits. The only en banc process available for Google, however, would have been at the Federal Circuit. It is understandable why Google chose to pursue a writ of certiorari at the Supreme Court rather than en banc review. It is unlikely that the Federal Circuit would have seen review of a unanimous panel decision interpreting regional circuit law as justifying the significant organizational resources of en banc review. Furthermore, such review could have jeopardized collegiality among Federal Circuit jurists on questions that are outside of the Federal Circuit's principal jurisprudence.

The Federal Circuit's expansive view of API copyrightability in conjunction with the jurisdictional misalignment has been costly for Google and the larger software industry. Google has now endured a second costly trial and has had to pursue its business and technology strategy under a cloud of confusion about the copyrightability of the functions and labels

---

ineligible for patent protection unless the claim entails inventive elements beyond any algorithms).

293. Forking of software code refers to creating an independent branch of a computer program. *See Fork (Software Development)*, WIKIPEDIA, https://en.wikipedia.org/wiki/Fork_(software_development) (last visited Oct. 8, 2017); Marcel T. Rosner & Andrew Kang, *Understanding and Regulating Twenty-First Century Payment Systems: The Ripple Case Study*, 114 MICH. L. REV. 649, 663 n.113 (2016). This split from the original program typically "spawns competing projects that cannot later exchange code, splitting the potential developer community." *See* Eric S. Raymond, *Promiscuous Theory, Puritan Practice*, *in* HOMESTEADING THE NOOSPHERE (2000), http://www.catb.org/~esr/writings/cathedral-bazaar/homesteading/ar01s03.html.

294. Atari Games Corp. v. Nintendo of Am., Inc., 897 F.2d 1572 (Fed. Cir. 1990).

within API packages. The greater software industry has endured continued uncertainty about the state of a critical aspect of copyright law.

There are several approaches to fix this bug in the appellate jurisdictional system. There is no justification for routing appeals of nonpatent issues governed by regional circuit law to the Federal Circuit when patent issues are not appealed. But were Congress to amend the Federal Circuit's jurisdiction so that such appeals would go to the regional circuit, patent owners could easily circumvent that rule by appealing patent issues they might otherwise drop solely to get the Federal Circuit to review nonpatent issues. More significantly, there are many patent cases with nonpatent issues that merit appeal on both patent and non–patent grounds. Hence, the allocation of appellate jurisdiction over cases raising patent and non–patent issues will arise.

Section A traces the legislative intent underlying the Federal Circuit's subject matter jurisdiction. Section B develops a framework for assessing appellate intellectual property jurisdiction. Section C applies that framework to assess appellate intellectual property jurisdictional regimes.

A. THE FEDERAL CIRCUIT'S SUBJECT MATTER JURISDICTION

The establishment of the Federal Circuit grew out of general concern about the federal judiciary's ability to keep pace with the demands of a growing nation, global economy, and ever–expanding and increasingly complex set of laws. Federal dockets had grown significantly in the 1960s and there was widespread concern about the strain on all levels of the federal judiciary.[295]

In 1972, Congress established the Commission on Revision of the Federal Court Appellate System, Structure and Internal Procedure to study the functioning of the appellate courts and make reform recommendations.[296] The Commission proposed, among other measures,

295. *See* PAUL D. CARRINGTON, AM. BAR FOUND., ACCOMMODATING THE WORKLOAD OF THE UNITED STATES COURTS OF APPEALS (1968); HENRY J. FRIENDLY, FEDERAL JURISDICTION: A GENERAL VIEW 31–47 (1973); COMM. ON RULES OF PRACTICE AND PROCEDURE, JUDICIAL CONFERENCE OF THE U. S., REPORT OF THE PROCEEDINGS OF THE JUDICIAL CONFERENCE OF THE UNITED STATES 38–39 (1971); Griffin B. Bell, *Toward a More Efficient Federal Appeals System*, 54 JUDICATURE 237, 237–38 (1971); Paul D. Carrington, *Crowded Dockets and the Courts of Appeals: The Threat to the Function of Review and the National Law*, 82 HARV. L. REV. 542 (1969).

296. *See* Roman L. Hruska, *The Commission on Revision of the Federal Court Appellate System: A Legislative History*, 1974 ARIZ. ST. L.J. 579 (1974).

the establishment of a National Court of Appeals.[297] The Commission also called attention to the problem of forum shopping in the patent field[298] but did not recommend creating a specialized court for patent appeals.[299] The Commission believed that its proposed National Court of Appeals would better address the patent forum shopping concerns. The Commission's grand appellate reform proposal, however, failed to gain passage in Congress.

Several years later, growing concerns about economic stagnation led President Carter's Domestic Policy Review on Industrial Innovation to pursue a specialized patent appellate court as a means of spurring research and development.[300] Advocates for a specialized patent appellate court believed that jurisprudential divisions among the regional courts of appeal undermined investment and innovative activity.[301] Many jurists, legislators, and key bar associations resisted the creation of a specialized patent tribunal, largely on the grounds that general jurists and regional courts best serve the administration of justice.[302] Supporters of consolidating patent

297. *See* HRUSKA COMMISSION REPORT, *supra* note 5.

298. *See id*. at 220–21 (quoting Judge Henry Friendly describing "mad and undignified races between a patentee who wishes to sue for infringement in one circuit believed to be benign toward patents, and a user who wants to obtain a declaration of invalidity or non-infringement in one believed to be hostile to them") (citing FRIENDLY, *supra* note 295).

299. *See id*. at 234–36.

300. *See* H.R. REP. No. 96-1307 (1980) (diagnosing the causes of economic stagnation as the "failure of American industry to keep pace with the increased productivity of foreign competitors"); Griffin B. Bell & Terence B. Adamson, *Daniel J. Meador—Visionary*, 80 VA. L. REV. 1209, 1212–13 (1994) (describing Daniel Meador's efforts as head of Office for Improvements in the Administration of Justice to establish the Federal Circuit); Helen W. Nies, *Special Session of the United States Court of Appeals for the Federal Circuit Commemorating Its First Ten Years*, 2 FED. CIR. B.J. 267, 270 (1992) ("Professor Meador was the first to conceive the idea of the Federal Circuit. As Assistant Attorney General from 1977 to 1979, he headed the Office for Improvements in the Administration of Justice which shepherded the legislation to create this court."); *see also generally* Elizabeth I. Rogers, *The Phoenix Precedents: The Unexpected Rebirth of Regional Circuit Jurisdiction over Patent Appeals and the Need for a Considered Congressional Response*, 16 HARV. J.L. & TECH. 411, 421–30 (2003).

301. *See* H.R. REP. No. 96-1307 (1980) (explaining that a single court for patent appeals "will do a great deal to improve investors' confidence in patented technology").

302. *See* George C. Beighley Jr., *The Court of Appeals for the Federal Circuit: Has it Fulfilled Congressional Expectations?*, FORDHAM INTELL. PROP. MEDIA & ENT. L.J. 670, 689–90, 693–97 (2011); *see also* S. REP. 97-275, at 40–41 (1981) (providing the additional views of Senator Max Baucus), *as reprinted in* 1982 U.S.C.C.A.N. 11, 50–51 ("[T]he American Bar Association and the American College of Trial Lawyers have actively opposed that portion of S. 1700 that would remove patent appeals jurisdiction from the eleven federal circuit courts of appeals. They share my concern that creating such a

appeals in a single tribunal countered that the proposed appellate tribunal, which merged appellate responsibilities for claims against the government, trade matters, and several other areas with appeals of patent cases, belied the "specialized court" label. The proposed court would have a range of responsibilities and include generalist judges.[303]

The counterargument carried the day. Congress passed the Federal Courts Improvement Act of 1981,[304] establishing a new Article III appellate tribunal: the U.S. Court of Appeals for the Federal Circuit. Nonetheless, legislators circumscribed the Federal Circuit's exclusive jurisdiction to preserve regional circuit court primacy in nonpatent areas of law.

Congress voiced concern about the Federal Circuit expanding its exclusive patent jurisdiction to other areas, such as antitrust law. The Senate Judiciary Committee noted the risk and specifically warned against the Federal Circuit's exclusive jurisdiction over patent claims being manipulated or extended. The Senate Report explained that the establishment of the Federal Circuit:

> is intended to alleviate the serious problems of forum shopping among the regional courts of appeals on patent claims by investing exclusive jurisdiction in one court of appeals. *It is not intended to create forum shopping opportunities between the Federal Circuit and the regional courts of appeals on other claims*.[305]

The Committee noted that:

> If, for example, a patent claim is manipulatively joined to an antitrust action, but severed or dismissed before final decision of the antitrust claim, jurisdiction over the appeal of the antitrust

specialty court is not in the best interest of the legal system."); Dennis Crouch, *An Open Letter from Judge Rader*, PATENTLY-O (June 30, 2014), http://patentlyo.com/patent/2014/06/letter-judge-rader.html (expressing regret in a farewell letter to his colleagues on the Federal Circuit that as a Senate Judiciary Committee staffer in the early 1980s, he "allowed judges from the Ninth Circuit to dissuade [him] from offering an amendment to include copyright and trademark cases within the jurisdiction of the Federal Circuit.").

303. *See* Daniel J. Meador, *Retrospective on the Federal Circuit: The First 20 Years—A Historical View*, 11 FED. CIR. B.J. 557, 558 (2001); Daniel J. Meador, *Origin of the Federal Circuit: A Personal Account*, 41 AM. U. L. REV. 581 (1992).

304. Federal Courts Improvement Act, Pub. L. No. 97-164, 96 Stat. 25 (1982).

305. *See* S. REP. 97-275 (1981), *as reprinted in* 1982 U.S.C.C.A.N. 11; *see also* H.R. REP. No. 96-1307, at 23 (1980) ("[J]urisdiction of an appeal in a case involving a claim arising under any Act of Congress relating to copy rights or trademarks . . . will continue to go to the regional appellate courts, pursuant to section 1294 of title 28").

> claim should not be changed by this Act but should rest with the regional court of appeals.[306]

Senator Leahy specifically warned that "[i]n nearly all . . . litigation [other than patent cases], science and technology, when relevant, are related to other human or social issues, and only a generalist court should ever hear such matters."[307]

Congress did not, however, clearly foresee the potential for jurisprudential confusion and forum shopping that could arise from the Federal Circuit's interpretation of regional circuit law. The potential for cases raising both patent and copyright questions would have seemed remote at the time that the Federal Circuit was crafted. Software litigation was in its infancy at the time, with substantial questions about the patent eligibility of computer software.[308] Even as software patenting expanded in the mid to late–1990s, the decline of software copyright litigation meant that complaints asserting both patent and copyright causes of action were rare.

The Federal Circuit's 2014 *Oracle v. Google* decision validates legislators' fears of overreach. The Federal Circuit's questionable interpretation of Ninth Circuit copyright law now motivates software intellectual property owners to bundle patent and copyright claims in order to take advantage of the Federal Circuit's expansive interpretation of software copyright protection. It is no coincidence that Cisco filed its complaint alleging software patent and copyright causes of action against Arista Networks after the Federal Circuit's 2014 *Oracle v. Google* decision.[309] The following Sections explore how the courts and Congress can ensure fidelity to regional circuit copyright law and prevent appellate forum shopping.

B. ANALYTICAL FRAMEWORK FOR ASSESSING APPELLATE INTELLECTUAL PROPERTY JURISDICTION

If we were assessing the design of intellectual property jurisdiction on a clean slate, the role for specialization and expertise would come

306. *See* S. REP. 97-275 (1981), *as reprinted in* 1982 U.S.C.C.A.N. 11.

307. *See* S. REP. 97-275, at Appendix B (1981), *as reprinted in* 1982 U.S.C.C.A.N. 11 (providing the additional views of Senator Patrick J. Leahy).

308. *See supra* note 292.

309. *See* Hardy, *supra* note 9; Graham, *supra* note 11.

centrally into play.[310] This Article, however, operates within the legislative landscape underlying the federal appellate system. There are four principal considerations within that constrained universe that guide the analysis of appellate jurisdiction of cases presenting both patent and copyright causes of action: jurisprudential integrity, federalism, specialization bias, and administrative efficiency.

1. Jurisprudential Integrity

The primary goal of appellate review is to ensure correct interpretation and application of the law. The traditional hierarchical nesting of district courts within regional circuits solves this problem through direct review of decisions by the regional circuit in which the district court sits. Intra–circuit splits can be addressed through en banc review. The Supreme Court provides a final judicial check and typically only intervenes to resolve inter–circuit splits.

Due to the divided appellate authority for patent and nonpatent issues, the federal judiciary comprises an overlap of appellate authority. By allocating exclusive jurisdiction to the Federal Circuit for patent appeals, the Federal Courts Improvement Act of 1981 created a form of surrogate appellate review of nonpatent issues. The Federal Circuit must interpret and apply the law of regional circuits in reviewing nonpatent issues. Such interpretive capacity is not uncommon within hierarchical judicial systems. For example, federal courts routinely interpret state and foreign law, and state courts sometimes interpret the law of other states.[311] Nonetheless, such surrogate review creates the potential for the emergence of potentially conflicting bodies of regional circuit law. If there is no effective means for checking the interpretive divergence, the integrity of regional jurisprudence is compromised.

While the interpretation and application of the law of another jurisdiction operates smoothly when the regional circuit law is settled, the task becomes difficult when the regional circuit law is inchoate, ambiguous,

310. *See, e.g.*, Laura G. Pedraza-Fariña, *The Federal Circuit: An Expert Community Approach*, 30 BERKELEY TECH. L.J. 89 (2015); Harold C. Wegner, *Federal Circuit Exclusive Appellate Patent Jurisdiction: A Response to Chief Judge Wood*, 13 CHI.-KENT J. INTELL. PROP. 394 (2014); Diane P. Wood, *Is It Time to Abolish the Federal Circuit's Exclusive Jurisdiction in Patent Cases?*, 13 CHI.-KENT J. INTELL. PROP. 1 (2013); Paul R. Gugliuzza, *The Federal Circuit as a Federal Court*, 54 WM. & MARY L. REV. 1791 (2013); Craig A. Nard & John F. Duffy, *Rethinking Patent Law's Uniformity Principle*, 101 NW. U. L. REV. 1619 (2007).

311. *See* Donald H. Zeigler, *Gazing into the Crystal Ball: Reflections on the Standards State Judges Should Use to Ascertain Federal Law*, 40 WM. & MARY L. REV. 1143 (1999).

or evolving. In such circumstances, Federal Circuit review of nonpatent issues creates a risk of divergent interpretation of regional circuit law.

2. Federalism

The division of responsibility among regional circuits reflects political struggles and compromises dating to the nation's founding. Deep divisions among the nation's founders hampered the establishment of a coherent intermediate appellate system for more than a century. At the nation's founding and continuing to some extent to this day, Federalists and Anti–Federalists divided over the extent of federal power. Federalists advocated a substantial national government and a strong lower federal judiciary. Anti–Federalists sought to weaken federal power, including judicial authority.[312] They advocated passage of a Bill of Rights to protect citizens against the tyranny of national government and preferred to leave judicial power within state government. The clash of perspectives played out in the First Congress in 1789, resulting in a grand compromise that produced the Bill of Rights and a limited system of lower federal courts tied to state boundaries.[313]

The 1789 Judiciary Act established three judicial levels—district courts, circuit courts, and, as set forth in the Constitution, the Supreme Court. The district courts and Supreme Court corresponded roughly to their modern forms. Each state had a single district court. District court jurisdiction, however, was limited and far narrower than the Constitution authorized. Congress authorized federal district courts to adjudicate admiralty, diversity of citizenship, federal criminal, and U.S. plaintiff cases. The original U.S. Supreme Court had a Chief Justice and five associate justices.

The early circuit courts, however, were very different from their contemporary counterparts. The jurisdiction of the circuit courts was limited to cases involving diversity of citizenship, major federal crimes, cases brought by the U.S. government, and larger civil and admiralty cases. The three circuit courts (one for northeastern districts, one for central Atlantic

312. *See generally* RUSSELL R. WHEELER & CYNTHIA HARRISON, CREATING THE FEDERAL JUDICIAL SYSTEM (3d ed. 2005) (tracing the history of the federal judiciary).

313. Reflecting the complexity and dynamism of the issues and the times, James Madison, an early Federalist and advocate for ratification of the United States Constitution as the foundation for a strong national power, *see* THE FEDERALIST NO. 10 (James Madison), broke with Alexander Hamilton and the Federalist Party in 1891 and organized the Democratic–Republican Party with Thomas Jefferson. He played a central role in drafting and ratifying the Bill of Rights, a cornerstone of the Anti–Federalists' effort to weaken national power.

states, and one for southern states) sat twice each year in one or two specified cities of each district. The circuit panel comprised two Supreme Court justices assigned to that circuit (hence the phrase "riding circuit") and the district judge in that district; initially, there was only one district judge authorized for each district (i.e., for each state).

As the United States' geographical reach and national economy developed, the jurisdiction and size of the federal judiciary grew.[314] The need for more effective judicial administration increased further as federal law expanded. For much of the first century of the United States judiciary, the circuit courts operated principally through Supreme Court justices "circuit riding" among the districts and hearing appeals in conjunction with district judges.[315]

In his first State of the Union message to Congress in 1861, President Lincoln declared, "the country has outgrown our present judicial system."[316] He noted that the eight recently admitted states had never had "circuit courts attended by supreme judges" and that adding enough justices to the Supreme Court to accommodate all the circuit courts that were needed would make the Supreme Court "altogether too numerous for a judicial body of any sort." Lincoln proposed fixing the Supreme Court at a "convenient number," irrespective of the number of circuits and dividing the country "into circuits of convenient size." The circuit courts could be served by either Supreme Court justices or judges appointed specifically for the circuit courts.

As the backlog of appeals grew, Congress eventually established nine circuit judgeships in 1869, far below the number needed to handle the mushrooming appellate backlog. The Judiciary Act of 1875 expanded federal jurisdiction to include federal questions and cases alleging more than $500 in controversy. Growing dockets and budgetary pressures

314. *See* WHEELER & HARRISON, *supra* note 312, at 9. Congress doubled the number of circuit courts in 1802, with one Supreme Court justice assigned to each circuit. Act of Apr. 29, 1802, 2 Stat. 118 (1802). As the number of states and territories expanded, necessitating additional district and circuit courts, Congress expanded the number of Supreme Court justices accordingly. In 1863, Congress created a tenth seat on the Supreme Court, Act of Mar. 3, 1863, 12 Stat. 794 (1863) ("To provide Circuit Courts for the Districts of California and Oregon . . . ."), although the full court rarely convened due to illness and vacancies. *See* Carl B. Swisher, *The Taney Period, 1836–64*, *in* 5 THE HISTORY OF THE SUPREME COURT OF THE UNITED STATES 839 (1974). Six years later Congress set the number of justices at nine, *see* Judiciary Act of 1869, 16 Stat. 44, where it has remained.

315. *See* WHEELER & HARRISON, *supra* note 312, at 7–19.

316. *See* Abraham Lincoln, *Message to Congress of Dec. 3, 1861*, *in* 5 THE COLLECTED WORKS OF ABRAHAM LINCOLN 41 (R. Basler ed., 1953).

strained the federal judiciary. Much of the burden fell to the 65–odd district judges, who were hearing close to 90 percent of the appeals by the 1880s, in addition to their large and growing trial court responsibilities.[317] Furthermore, the Supreme Court was obliged to hear almost all cases in which litigants sought high court review, resulting in a massive logjam at the top of the federal judiciary pyramid.[318]

Thus, a century after its establishment, the federal judiciary was in crisis.[319] Supreme Court justices had long abandoned riding circuit. The ranks of intermediate circuit judgeships were inadequate to handle the rising appellate caseload, adding substantial additional burden to an overextended district judge corps. Moreover, broad access to the Supreme Court impeded its capacity to review cases with alacrity.

Dissatisfaction with the operation of the federal judiciary ultimately led Congress to pass the Circuit Court of Appeals Act of 1891,[320] commonly referred to as the Evarts Act, establishing the modern circuit court system. Senator Evarts orchestrated a compromise that increased the role of the national courts while preserving state and regional influence.[321] Furthermore, the legislation substantially shifted the Supreme Court's workload to nine separate regional circuit courts of appeals and authorized the appointment of 19 circuit court judges, three for the Second Circuit and two for each of the others.

The relatively small scale of appellate courts—initially 19 circuit judges among the nine regional circuits—functioned smoothly during the first several decades, leaving the Supreme Court to focus on inter–circuit splits. As Congress expanded the size of the appellate judiciary to address growing caseloads, the problem of intra–circuit splits emerged. The Supreme Court addressed these issues by authorizing en banc review of intra–circuit

317. *See* FELIX FRANKFURTER & JAMES LANDIS, THE BUSINESS OF THE SUPREME COURT 60, 79 (1928) (reporting that the number of cases pending in the federal courts rose 86%—from 29,000 to 54,000—between 1873 and 1890).

318. By 1890, the Supreme Court had 1,816 cases on its docket, including 623 cases filed that year. *See id.* at 101–02.

319. *See* PAUL M. BATOR ET AL., HART & WECHSLER'S THE FEDERAL COURTS AND THE FEDERAL SYSTEM 37 (3d ed. 1988) (referring to the post–Civil War period as "the nadir of federal judicial administration").

320. *See* Evarts Act, Act of Mar. 3, 1891, 26 Stat. 826 (1891); WHEELER & HARRISON, *supra* note 312, at 16–18; *See* ERWIN C. SURRENCY, HISTORY OF THE FEDERAL COURTS 49 (1987).

321. *See* WHEELER & HARRISON, *supra* note 312, at 16–18.

conflicts.[322] Congress codified the Supreme Court's *Textile Mills* decision in passing the Judicial Code of 1948.[323]

Congress's decision to centralize and consolidate appellate patent review consciously diverged from the federalist structure of the appellate courts, for the purpose of eliminating the interpretive confusion and forum shopping that had emerged in patent cases. Yet, federalism concerns were voiced during consideration of the Federal Courts Improvement Act[324] and Congress retained the federalist judicial structure for non–patent issues. The failure to provide a mechanism to ensure fidelity to regional circuit law, however, creates the potential for the confusion manifested in the *Oracle v. Google* litigation.

---

322. *See* Textile Mills Sec. Corp. v. Comm'r, 314 U.S. 326 (1941) (interpreting the Judicial Code to permit en banc review). Congress supplanted the Evarts Act by passing the Judicial Code in 1911. *See* Act of Mar. 3, 1911, ch. 231, 36 Stat. 1087.

323. Section 46(c) of the Judicial Code of 1948 provided that circuit courts could convene en banc panels upon a majority vote of active judges in the circuit. "A court in banc shall consist of all active circuit judges of the circuit." Act of June 25, 1948, ch. 646, 62 Stat. 871 (1948). Congress left the specific procedures and standards for doing so, however, to the circuit courts. *See* Pub. L. No. 88-176, 77 Stat. 331 (1963).

324. *See, e.g.*, S. REP. 97-275, at 40–41 (1981) (quoting HRUSKA COMMISSION REPORT, *supra* note 5):

> Giving a national court exclusive jurisdiction over appeals in a category of cases now heard by the circuit courts would tend to dilute or eliminate regional influence in the decision of those cases. Our nation is not yet so homogenous that the diversity of our people cannot be reflected to some advantage in the decisions of the regional courts. Excluding these courts from consideration of particular categories of cases would also contract the breadth of experience and knowledge which the circuit judges would bring to bear on other cases; the advantages of decision making by generalist judges diminish as the judge's exposure to varied areas of the law is lessened.);

Id.; *see also id.* (additional Views of Senator Max Baucus) ("Many of us in the Congress have been greatly disturbed by the growing trend toward centralizing decision making in Washington, D.C. Many of us have supported venue reform to ensure that cases are litigated in States, where the problems arise, rather than in the District of Columbia. Similarly, I believe that we must avoid centralized specialty courts."); S. REP. 97-275, at Appendix B (1981), *as reprinted in* 1982 U.S.C.C.A.N. 11 (additional Views of Senator Patrick J. Leahy on S. 1700) (advocating creation of the Federal Circuit, but agreeing with "the concerns expressed about the precedent of establishing specialty courts, which in general would be very detrimental to our tradition of diversity and independence on the bench").

3. Specialization Bias

Political scientists, legal scholars, and jurists have long worried that specialized courts are more prone to political influences[325] and tunnel vision[326] than courts of general jurisdiction.[327] The legislative record shows that corporate interests played a large role in creating the Federal Circuit.[328] In a study of the Court of Customs and Patent Appeals (CCPA), one of the courts merged into the Federal Circuit that handled appeals of patent examination, Professor Lawrence Baum found that:

> [t]he patent specialists on the court, appointed through the efforts of the patent bar, have led the CCPA to adopt a line of policy significantly different from the patent policies that prevail in most of the federal judiciary. The CCPA's specialization ultimately has

325. *See* H.R. REP. No. 97-312, at 31 (1981) (noting that "[s]everal witnesses . . . expressed fears that the Court of Appeals for the Federal Circuit would be unduly specialized or would soon be captured by specialized interests"); *see also* Lawrence Baum, *Judicial Specialization, Litigant Influence, and Substantive Policy: The Court of Customs and Patent Appeals*, 11 LAW & SOC'Y REV. 823 (1977) (arguing that court specialization enhances the likelihood of litigant interest groups affecting substantive policy).

326. *See* Rochelle Cooper Dreyfuss, *The Federal Circuit: A Case Study in Specialized Courts*, 64 N.Y.U. L. REV. 1, 3 (1989); Alan B. Parker, *Examining Distinctive Jurisprudence in the Federal Circuit: Consequences of a Specialized Court*, 3 AKRON INTELL. PROP. J. 269, 287–89 (2009) (discussing concerns of doctrinal, intellectual, and judicial isolation); Simon Rifkind, *A Special Court for Patent Litigation? The Danger of a Specialized Judiciary*, 37 A.B.A. J. 425, 425–26 (1951) (expressing concern that the specialization and "seclusiveness" of patent law "immunizes it against the refreshment of new ideas, suggestions, adjustments and compromises which constitute the very tissue of any living system of law").

327. *See generally* WILLIAM M. LANDES & RICHARD A. POSNER, THE POLITICAL ECONOMY OF INTELLECTUAL PROPERTY LAW 26–27 (2004) (arguing that "a specialized court is more likely to have a 'mission' orientation than a generalist court"); William M. Landes & Richard A. Posner, *An Empirical Analysis of the Patent Court*, 71 U. CHI. L. REV. 111, 111–12 (2003) (positing that a specialized patent court is more likely than a generalist court to take a strong stance on its subject matter because "interest groups that had a stake in patent policy would be bound to play a larger role in the appointment of the judges of such a court than they would in the case of the generalist federal courts").

328. *See* LAWRENCE BAUM, SPECIALIZING THE COURTS 181, 204 (2011) (noting that corporate support played a key role in creation of the Federal Circuit); Paul R. Gugliuzza, *Rethinking Federal Circuit Jurisdiction*, 100 GEO. L.J. 1437, 1458 (2012); F.M. Scherer, *The Political Economy of Patent Policy Reform in the United States*, 7 J. TELECOMM. & HIGH TECH. L. 167, 190 (2009) (noting the strong support from corporate patent counsel).

> been responsible for the court's distinctive path in the past two decades.[329]

The legislation creating the Federal Circuit as well as the initial appointment of jurists from the CCPA implemented a mission of strengthening the patent system through statutory interpretation and evolution of non–statutory patent doctrines.[330] This mission has been reinforced through the emergence of a dedicated, well–funded bar and numerous patent–focused industry organizations.[331] While such organizations produce valuable research and education, it would be naive to think that the ecosystem surrounding patent adjudication did not promote the agenda of the most active and interested constituencies.[332]

Academic research finds that the Federal Circuit has strengthened patent protection through statutory interpretation and evolution of non–statutory patent doctrines. Multiple scholars have chronicled particular doctrinal patterns (such as formalism and textualism) that strengthen patent

---

329. *See* Lawrence Baum, *Judicial Specialization, Litigant Influence, and Substantive Policy: The Court of Customs and Patent Appeals*, 11 LAW & SOC'Y REV. 823, 845–46 (1977).

330. Gugliuzza, *supra* note 328, at 1458 (discussing strong industrial support for creating the U.S. Court of Appeals for the Federal Circuit); LANDES & POSNER, *supra* note 327, at 26–27 (noting that the Federal Circuit "has defined its mission as promoting technological progress by enlarging patent rights").

331. The Federal Circuit Bar Association, founded in 1985, "unites the various groups who practice within the Circuit community, including the private and public sectors and litigators as well as agency and house counsel." *See Mission & Vision*, FED. CIRCUIT BAR ASS'N, https://fedcirbar.org/About-FCBA/Who-We-Are/Mission-Vision (last visited Oct. 8, 2017) The American Bar Association, intellectual property owners, pharmaceutical industry, and high technology industries have long had strong advocacy arms. The biotechnology and software industries have become increasingly active. Not all of these constituencies favor strong patent rights, which produced a more complex political dynamic during the past two decades and the lead–up to the America Invents Act. Nonetheless, the Federal Circuit has remained focused on strong patent rights and a robust patent system.

332. *See* LANDES & POSNER, *supra* note 327, at 26–27 (suggesting that Federal Circuit favors increased demand for the services of its primary constituency, patent lawyers); *see generally* William N. Eskridge, Jr., *Politics Without Romance: Implications of Public Choice Theory for Statutory Interpretation*, 74 VA. L. REV. 275 (1988).

protection.[333] Empirical research finds that the Federal Circuit views patent holders more favorably than regional circuit courts.[334]

The Federal Circuit, however, has not always expanded the scope of patent protection. For example, the Federal Circuit's formalism led the court to cabin the doctrine of equivalents.[335] More recently, the Federal Circuit has reined in patent damage theories.[336] On balance, however, the Federal Circuit has favored broad patentability,[337] narrow limitations,[338] and robust appellate authority.[339]

These tendencies raise the concern that the Federal Circuit would favor a broader scope of copyright protection for computer software than regional circuit courts. The Federal Circuit's 2014 *Oracle v. Google* decision appears to bear this out. The Federal Circuit downplayed the legislative concern for ensuring that copyright protection for computer software does not extend to functional features, which is the province of patent protection. Moreover, the court read the Ninth Circuit's jurisprudence, particularly the *Sega*

333. *See* Dreyfuss, *supra* note 326; Peter Lee, *Patent Law and the Two Cultures*, 120 Yale L.J. 2 (2010) (arguing that the Federal Circuit's formalism disengages from technology); Arti K. Rai, *Engaging Facts and Policy: A Multi-Institutional Approach to Patent System Reform*, 103 COLUM. L. REV. 1035, 1103–14 (2003) (suggesting that formalism might mask bias); John R. Thomas, *Formalism at the Federal Circuit*, 52 AM. U. L. REV. 771 (2003) (cautioning that the Federal Circuit should look beyond certainty and predictability in developing legal rules).

334. *See* John R. Allison & Mark A. Lemley, *Empirical Evidence on the Validity of Litigated Patents*, 26 AIPLA Q.J. 185, 205–06 (1998); Matthew D. Henry & John L. Turner, *The Court of Appeals for the Federal Circuit's Impact on Patent Litigation*, 35 J. LEGAL STUD. 85, 114 (2006); Glynn S. Lunney, Jr., *Patent Law, the Federal Circuit, and the Supreme Court: A Quiet Revolution*, 11 SUP. CT. ECON. REV. 1, 15–16 (2004).

335. *See* John R. Thomas, *Formalism at the Federal Circuit*, 52 AM. U. L. REV. 771, 772–75 (2003); *cf.* Festo Corp. v. Shoketsu Kinzoku Kogyo Kabushiki Co., 535 U.S. 722, 738 (2002) (rejecting the complete bar to the doctrine of equivalents and noting that "we have consistently applied the doctrine [of equivalents] in a flexible way, not a rigid one").

336. *See* VirnetX, Inc. v. Cisco Sys., Inc., 767 F.3d 1308 (Fed. Cir. 2014); LaserDynamics Inc. v. Quanta Comput., Inc., 694 F.3d 51 (Fed. Cir. 2012); Uniloc USA, Inc. v. Microsoft Corp., 632 F. 3d 1292 (Fed. Cir. 2011).

337. *See, e.g.* Biosig Instruments, Inc. v. Nautilus, Inc., 715 F.3d 891 (Fed. Cir. 2014), *vacated,* Nautilus, Inc. v. Biosig Instruments, Inc., 134 S. Ct. 2120 (2014); MercExchange, L.L.C. v. eBay, Inc., 401 F.3d 1323 (Fed. Cir. 2005), *rev'd*, eBay, Inc. v. MercExchange, L.L.C., 547 U.S. 388 (2006); Teleflex Inc. v. KSR Int'l Co., 119 Fed. Appx. 282 (Fed. Cir. 2005), *rev'd*, KSR Int'l Co., v. Teleflex Inc., 550 U.S. 398 (2007); State St. Bank & Tr. Co. v. Signature Fin. Grp., Inc., 149 F.3d 1368 (Fed. Cir. 1998).

338. *See, e.g.,* Madey v. Duke Univ., 307 F.3d 1351, 1360–61 (Fed. Cir. 2002) (construing the common law experimental use defense narrowly).

339. *See* Teva Pharm. USA, Inc. v. Sandoz, Inc., 723 F.3d 1363 (Fed. Cir. 2013), *vacated*, 135 S. Ct. 831 (2015).

decision, narrowly. Furthermore, the court placed great emphasis on its own application of Ninth Circuit law in the *Atari Games* case.[340]

It is perhaps surprising that the Federal Circuit's formalism and experience with patent law did not push the court toward a narrower scope of copyright protection. As the panel recognized, the fair use doctrine is the most unpredictable doctrine in copyright law.[341] Furthermore, broad copyright protection for computer software impinges on patent law's primacy in promoting technological advance. *Baker v. Selden* preempts copyright protection of functional elements and methods of operation. Nonetheless, protecting intellectual property carried the day. It is reasonable to believe that like the First Circuit in *Lotus v. Borland*, the Ninth Circuit would have taken greater cognizance of the anticompetitive concerns of broad intellectual property protection. The legislative history of the Federal Courts Improvement Act highlighted the concern that a patent–centric specialty court might be less sensitive to antitrust policy.[342]

4. Administrative Efficiency

Administrative efficiency weighs against regional circuit jurisprudential integrity, federalism, and specialization–bias considerations in allocating appellate jurisdiction.[343] Various doctrines promote bundling causes of action arising out of the same transaction or occurrence to prevent piecemeal litigation.[344] There may be administrative efficiency reasons to bundle the appeal of those causes of action as well, but such efficiencies might be relatively modest and in tension with other jurisdictional considerations.

340. *See Oracle II*, 750 F.3d at 1357, 1360, 1361, 1363, 1366, 1370 (citing Atari Games Corp. v. Nintendo of Am., Inc., 897 F.2d 1572 (Fed. Cir. 1990) as substantive copyright law authority).

341. *See Oracle II*, 750 F.3d at 1372 (quoting Monge v. Maya Magazines, Inc., 688 F.3d 1164, 1170 (9th Cir. 2012) (quoting Dellar v. Samuel Goldwyn, Inc., 104 F.2d 661, 662 (2d Cir. 1939) (per curiam)).

342. *See supra* notes 305–307 and accompanying text.

343. Congress considered administrative efficiencies in establishing the Federal Circuit. *See* Daniel J. Meador, *An Appellate Court Dilemma and a Solution Through Subject Matter Organization*, 16 U. MICH. J.L. REFORM 471 (1983).

344. *See* RESTATEMENT (SECOND) OF JUDGMENTS § 24(1) (AM. LAW INST. 1982) ("When a valid and final judgment rendered in an action extinguishes the plaintiff's claim . . . the claim extinguished includes all rights of the plaintiff to remedies against the defendant with respect to all or any part of the transaction, or series of connected transactions, out of which the action arose."); *see generally* CHARLES A. WRIGHT & ARTHUR R. MILLER, FEDERAL PRACTICE & PROCEDURE §§ 4401–09 (3d ed. 2002).

District judges have substantial flexibility in managing litigation, including staging and bifurcation.[345] Judge Alsup sensibly phased the copyright and patent causes of action in *Oracle v. Google*, with both sets of issues tried to the same jury. This will generally be sound case management due to the substantial differences between patent and copyright law.

Given that Oracle did not appeal dismissal of the patent causes of action, there is no administrative efficiency (or other) basis for the Federal Circuit, as opposed to the Ninth Circuit, to hear the *Oracle v. Google* appeal. Speculatively inferring what a Ninth Circuit panel would do could have been avoided. Moreover, appeal to the Ninth Circuit would have assured regional circuit jurisprudential integrity. Furthermore, it is not clear that dividing the case along patent and nonpatent causes of action where both sets of issues are appealed would add substantial complexity. The next Section explores these scenarios

C. REFINING APPELLATE INTELLECTUAL PROPERTY JURISDICTION

The challenge for jurists and policymakers is promoting faithful interpretation of regional circuit nonpatent jurisprudence without jeopardizing administrative efficiency. The development of a distinct line of Federal Circuit interpretation of regional circuit jurisprudence creates the potential for a new form of forum shopping. By pursuing patent claims with nonpatent causes of action, parties can opt into the Federal Circuit's interpretation of regional circuit law. That might be advantageous where an intellectual property owner seeks a more expansive interpretation of intellectual property protection. Moreover, because of the lack of error correction short of Supreme Court review, the jurisprudential structure of the federal appellate judiciary will be distorted. Contrary to Congress's intent, the Federal Circuit can effectively override regional circuits' nonpatent bodies of law.

This Section proposes restructuring appellate intellectual property jurisdiction along case management lines. Subsection 1 analyzes case management options for district courts. Subsection 2 analyzes appellate case management reforms.

1. District Court Case Management and Routing of Appellate Review

The core appellate jurisdiction problem traces back to the filing of a complaint asserting patent and non–patent causes of action. Once a plaintiff

345. *See generally* PETER S. MENELL ET AL., PATENT CASE MANAGEMENT JUDICIAL GUIDE (3d ed. 2016).

files a complaint containing a patent cause of action, Section 1295(a)(1) of Title 28 confers exclusive appellate jurisdiction on the Federal Circuit. Yet Congress seeks to have regional circuit law (or possibly state law) apply to the nonpatent causes of action. By starting with district court case management, various opportunities to promote fidelity to regional circuit can be pursued while preserving the Federal Circuit's authority to interpret and apply federal patent law.

Although the patent and nonpatent issues may share a common nucleus of operative facts, the applicable legal standard may be sufficiently distinct that it makes sense to phase or bifurcate trial of the causes of action, as occurred in the *Oracle v. Google* case. To the extent that the district court keeps the trial and post–trial rulings separate, it is as if separate cases have been adjudicated.

If exclusively patent or nonpatent issues are appealed, jurisdictional integrity and federalism considerations favor having those issues resolved by the appellate tribunal with primary authority: the Federal Circuit for patent issues and the regional circuit court for the nonpatent issues. Thus, since there is no loss in administrative efficiency for cases in which patent issues are not appealed, the most obvious solution would be to vest jurisdiction over the appeal of the nonpatent issues with the regional appellate court. This could be accomplished by amending § 1295(a)(1) of Title 28 of the U.S. Code to exclude from the Federal Circuit's appellate jurisdiction cases that do not appeal issues arising under the Patent Act or Plant Variety Protection Act.

If both patent and nonpatent issues are appealed, the only difference would be that the case would be effectively divided into separate causes of action and the timing of appeals and remands could affect case management. But since the case was already phased or bifurcated, appellate bifurcation would be straightforward and not add significant additional administrative cost. The district court would retain jurisdiction and could adapt any further proceedings based on the outcome and timing of the parallel appellate processes. Section 1295(a) could be amended to provide for cases in which the nonpatent issues have been tried separately—whether through phasing or bifurcation—to fall within the appellate jurisdiction of the regional circuit court of appeals.

That leaves cases in which the patent and nonpatent issues have been litigated in a combined proceeding and are intertwined. This might occur, for example, in cases involving patent claims and antitrust counterclaims. Even in such scenarios, the appeal could be best handled by the regional circuit court if no patent issues are appealed. If patent issues or patent and

interrelated, nonpatent issues are appealed, then the Federal Circuit has primacy in adjudicating the appeal.

2. Appellate Jurisdiction Reforms

Even where the Federal Circuit considers regional circuit law questions, there are several opportunities to improve fidelity to regional circuit jurisprudence. Section (a) considers implementation of a system analogous to the certification of state law questions to the highest state court tribunal. Section (b) considers adding a second tier of appellate review at the regional circuit level. Section (c) discusses ramifications for the Supreme Court's role.

a) Certification of Questions to Regional Circuit Courts

The federal courts have long dealt with the interpretation and application of other bodies of law. Following the Supreme Court's decision in *Erie Railroad Co. v. Tompkins*[346] largely eliminating general federal common law, federal courts have had to apply state law in diversity jurisdiction cases. Beginning in the 1960s, most states have afforded federal courts the option of certifying questions of state law to the highest court in the state.[347] The federal court can go directly to the highest state court to resolve difficult interpretive questions. This process, however, is at the discretion of the federal court.

Although Congress could authorize an analogous process for the Federal Circuit to certify complex questions of regional circuit law to the regional circuit court, such a process would be unduly cumbersome. Unlike the highest court in a state, regional circuits typically sit in panels smaller than the full bench. En banc review is a relatively infrequent process. It is also relatively complicated to organize and it can take a long time to render decisions on complex issues. Furthermore, it might be difficult to boil down a question of regional circuit copyright law or other nonpatent issue to a clear question that can easily be applied. A better approach would be to develop a mechanism for direct review of the Federal Circuit's nonpatent issues at the regional circuit level.

b) Regional Circuit Review

Even if the nonpatent issues cannot be separated from the patent issues prior to the first appeal level, Congress could provide for Federal Circuit

346. Erie R.R. Co. v. Tompkins, 304 U.S. 64 (1938).

347. *See* Rebecca A. Cochran, *Federal Court Certification of Questions of State Law to State Courts: A Theoretical and Empirical Study*, 29 J. LEGIS. 157, 159 n.13 (2003).

interpretations of regional circuit review to create an additional layer of appellate review by a regional circuit panel and/or at the en banc level. Thus, Congress could provide for an optional second level of appellate review. A party that believed that the Federal Circuit has misinterpreted or misapplied regional circuit jurisprudence could challenge that decision through a second–panel review within the regional circuit. Alternatively, a party challenging the Federal Circuit's interpretation of regional circuit law could file an en banc petition in the regional circuit.

These approaches provide a sensible and balanced solution to the regional circuit jurisprudential integrity and forum shopping problems while avoiding undue administrative costs. Given the Supreme Court's severe capacity constraints[348] and disinclination to consider interlocutory appeals[349] and intra–circuit splits,[350] providing litigants a regional circuit review option could provide a valuable secondary screen to ensure fidelity to regional circuit authority. It could also avoid the additional costs from Federal Circuit remands on unnecessary issues. On the cost side of the equation, an additional appellate review would add further time to resolving disputes. But in cases like *Oracle v. Google*, such an option would have potentially avoided a costly second trial and would likely have provided a clear answer to core questions about API copyrightability in the Ninth Circuit.

A regional circuit appeal process would also conserve Supreme Court resources. When the Federal Circuit misinterprets regional circuit law, it effectively creates an intra–circuit split. If Google had the option to pursue regional circuit review, it could have avoided filing its interlocutory certiorari petition. The copyright issues would have found their path within the regional circuit process. And only if one of the parties could allege an inter–circuit split would the case become ripe for Supreme Court review.

Whether to limit the second–level, regional appeal to the panel or en banc level would depend on an analysis of process costs and delay. Limiting a Federal Circuit litigant only to regional circuit en banc review of nonpatent issues would save resources, but would likely result in more Federal Circuit mutation of regional circuit law.

---

348. *See* Ryan J. Owens & David A. Simon, *Explaining the Supreme Court's Shrinking Docket*, 53 WM. & MARY L. REV. 1219 (2012); Kenneth W. Starr, *The Supreme Court and Its Shrinking Docket: The Ghost of William Howard Taft*, 90 MINN. L. REV. 1363 (2006); HRUSKA COMMISSION REPORT, *supra* note 5, at 209–14.

349. *See* SUP. CT. R. 11.

350. *See* SUP. CT. R. 10; Textile Mills Sec. Corp. v. Comm'r, 314 U.S. 326, 344–45 (1941).

c) Supreme Court Review

If Congress does not act to route appeals of separable nonpatent issues to regional circuits and provide an additional layer of circuit review, either panel or en banc, for nonpatent issues remaining after Federal Circuit review, then the Supreme Court should expand its certiorari criteria to consider Federal Circuit misinterpretation of regional circuit law. As the *Oracle v. Google* litigation highlights, it is possible that the copyrightability of APIs might never be ripe for Supreme Court review. The detour into a fair use trial has submerged the API copyrightability issue, which is of great importance to a substantial portion of the software industry.

VI. CONCLUSIONS

The Federal Circuit's exclusive jurisdiction over patent appeals, in conjunction with its questionable interpretation of Ninth Circuit copyright law, has produced a new class of forum shopping. By combining a patent cause of action with a software copyright cause of action in a district court filing anywhere within the boundaries of the Ninth Circuit, a plaintiff can opt for a far more expansive version of copyright protection than they could obtain if they only pursued the copyright cause of action. Congress specifically warned against the extension of the Federal Circuit's exclusive jurisdiction over patent appeals to nonpatent causes of action governed by regional circuit law in establishing the Federal Circuit.

Fortunately, some fairly straightforward adjustments to the Federal Circuit's jurisdiction can nip this problem in the bud. Congress can largely rectify the problem by providing for nonpatent issues resolved in separable proceedings to be appealed directly to the regional circuit court. Furthermore, Congress can address the problem presented by intertwined patent and nonpatent issues by providing for a second level of appellate review of the nonpatent issues in the regional circuit.

THE *WILLIAMSON* REVOLUTION IN SOFTWARE'S STRUCTURE

Kevin Emerson Collins[†]

ABSTRACT

In *Williamson v. Citrix Online*, the Federal Circuit altered the threshold test for determining whether a functional claim limitation that does not use the term "means" is governed by the scope–narrowing rules of § 112(f). Before *Williamson*, there was a strong presumption that functional claim limitations without the word "means" were not governed by § 112(f). After *Williamson*, § 112(f) now governs all functional limitations without the word "means" that do not recite sufficient structure for performing the claimed function.

A common, incrementalist interpretation of *Williamson* is that the alteration of the § 112(f) threshold test will only have a small impact on the law of functional claiming. This interpretation frames *Williamson* as a case that will simply move the needle on the *quantitative question* about structure in the threshold test: How much structure need be recited in a limitation to avoid § 112(f)?

In contrast, this Essay argues that, at least for software limitations in particular, *Williamson* will have a revolutionary impact on the law of functional claiming. *Williamson* will force the Federal Circuit to formulate a new answer to the more fundamental *definitional question* about software's structure: What constitutes structure in a software invention in the first place? *Williamson* demands a revolution in the definition of software's structure because the Federal Circuit's pre–*Williamson* doctrine that software's structure is an algorithm cannot do the work that *Williamson* requires. Although the concept of an algorithm can define corresponding structure in a specification in the course of construing the scope of a limitation that is known to be governed by § 112(f), it cannot identify structure in a claim limitation in the course of the threshold determination of whether or not a limitation is governed by § 112(f). After demonstrating that *Williamson* demands a revolution in software's structure under § 112(f), this Essay briefly also considers several different paths that the *Williamson* revolution could take.

DOI: https://dx.doi.org/10.15779/Z38SJ19R03

† Professor of Law, Washington University.

TABLE OF CONTENTS

I. INTRODUCTION

In *Williamson v. Citrix Online*, the Federal Circuit altered the threshold test for determining whether a limitation that employs functional language is subject to § 112(f) of the Patent Act.[1] Section 112(f) expressly sanctions broad, functional claim language: claim limitations "may be expressed as a means . . . for performing a specified function without the recital of structure [or] material . . . in support thereof."[2] However, as a price for using such language, § 112(f) mandates a scope–narrowing rule of claim construction: functional limitations "shall be construed to cover [only] the corresponding structure [or] material . . . described in the specification and equivalents thereof."[3] In the decade preceding *Williamson*, the Federal Circuit developed formalistic, strong presumptions that usually allowed a patent

1. Williamson v. Citrix Online, LLC, 792 F.3d 1339, 1347–49 (Fed. Cir. 2015) (en banc).
2. 35 U.S.C. § 112(f) (2012).
3. *Id.*

drafter to opt into or, more importantly, out of § 112(f). If a functional claim limitation contained the word "means," § 112(f) almost always governed. However, if the limitation employed a synonym for "means," such as "device" or "mechanism"—words that the Federal Circuit refers to as "nonce" words—§ 112(f) almost never applied.[4] *Williamson* expanded the reach of § 112(f) by reducing the strength of the latter presumption and allowing it to be overcome with a showing that a limitation recites "function without reciting sufficient structure for performing that function."[5] In short, *Williamson* promises to rein in the scope of some overbroad, functionally defined claims that do not employ the term "means."

A year after *Williamson*, there is little evidence of how the Federal Circuit will interpret its new § 112(f) threshold test.[6] One common prediction about *Williamson*'s future impact is that it may lead to only incremental change in the law of functional claiming. This view frames *Williamson* as a case that will do nothing but move the needle a bit on the *quantitative question* about structure in the threshold test: How much structure in a functional claim limitation is enough to avoid § 112(f)?[7] In contrast, this Essay argues that, with respect to software patents in particular, *Williamson* mandates a revolution in the law of functional claiming.[8] To bring *Williamson* to bear on software patents, the Federal

4. *Williamson*, 792 F.3d at 1348–49.

5. *Id.* at 1349 (citing Watts v. XL Sys., Inc., 232 F.3d 877, 880 (Fed. Cir. 2000)).

6. The Federal Circuit has decided only one case addressing the § 112(f) threshold test since *Williamson*. Media Rights Tech. v. Capital One Fin. Corp., 800 F.3d 1366, 1373 (Fed. Cir. 2015) (holding that a "compliance mechanism" limitation is subject to § 112(f)). For an overview of several district court cases in which the § 112(f) threshold test was performed both before *Williamson* and again after, see Paul R. Gugliuzza, *Early Filing and Functional Claiming*, 96 B.U. L. REV. 1223, 1232–43 (2016).

7. Gugliuzza, *supra* note 6; Luiz Felipe Oliveira, *Means-Plus-Function Claims: An Analysis of the Federal Circuit's Ruling in* Williamson v. Citrix Online, 11 J. INTELL. PROP. L. & PRACTICE 199 (2016); Eric P. Raciti, *Means Plus Function Claiming: What Does It Mean to Be a Means, When Are Means Means, and Other Meaningful Questions*, LANDSLIDE, March–April 2016, at 19. Incrementalists who assert that *Williamson*'s impact is limited to the quantitative question of the threshold test may, in theory, predict either minor or significant movement of the needle.

8. Software patents are widely recognized as troubling for a number of policy reasons, and the expansive scope that can be achieved through unbridled functional claiming has been identified as one of these reasons. Kevin Emerson Collins, *Patent Law's Functionality Malfunction and the Problem of Overbroad, Functional Software Patents*, 90 WASH. U. L. REV. 1399 (2013); Mark A. Lemley, *Software Patents and the Return of Functional Claiming*, 2013 WIS. L. REV. 905 (2013). Although there are good policy reasons to want to see *Williamson* lead to a revolution that cuts back on the permissible scope of functional software claims, this Essay does not enter the policy debate over

Circuit must ask and answer a fundamental question that, to date, it has not yet openly addressed. This fundamental question is the *definitional question* about software's structure. What properties of a software program should count as structure when they are recited as claim limitations? In other words, what is the structure of a functional software limitation in the first place?

To support the argument that *Williamson* mandates a revolution in the definition of software's structure, this Essay makes three cumulative points. First, it highlights the sui generis nature of the definitional question about § 112(f) structure in software. Identifying structure may be an intuitive exercise in most technologies, but it is not in software. Second, this Essay argues that the Federal Circuit ducked the definitional question about software's structure in its pre–*Williamson* jurisprudence. The Federal Circuit did define software's structure as an algorithm in the context of the search for corresponding structure in the specification during claim construction, but an algorithm provides only a relational definition that is meaningless in the search for structure in a claim's limitations during the § 112(f) threshold test. Third, this Essay maps out the difficult path forward that the Federal Circuit must follow in the post–*Williamson* era in order to provide a meaningful answer to the definitional question about software's structure in the § 112(f) threshold test.

Software is an unusual technology. For most technologies, including mechanical technologies, the answer to the definitional "What is structure?" question is intuitive to technological neophytes and experienced patent judges alike: structure includes the physical, spatial, and material properties of a technology. However, in software, the definitional question does not have a simple answer. The physical, structural qualities of a software invention are irrelevant to the definition of what a software inventor has actually invented. Software has been engineered with the express goal of allowing programmers to remain ignorant of the physical structure of their inventions. There is no relevant physical structure on which an economically rational patent regime can rely to curtail the scope of functional claims; instead, software inventions can only reasonably be defined by reciting what the software does or how it performs in functional terms. Thus, a sui generis definition of structure is needed to make § 112(f) a meaningful limit on software claims. This definition must invoke software's metaphorical or logical structure in the sense that a more granular, functional description of how software performs should count as

---

functional claiming. It more modestly reveals the conceptual poverty of the pre–*Williamson* definition of software's structure and the need that *Williamson* creates to rethink that definition.

a description of structure, despite the fact that the description is still functional as a purely linguistic matter.

To be clear, the idea that § 112(f) requires a sui generis definition of software's structure had been recognized well before *Williamson*. In the context of searching for the corresponding structure of a § 112(f) limitation in the specification during claim construction, the Federal Circuit has long held that software's structure is an "algorithm" or a step–by–step procedure specifying how to perform the function recited as a claim limitation.[9] Given that the Federal Circuit has already answered the definitional question in this context, the incrementalist prediction about the likely impact of *Williamson* might seem eminently reasonable, even with respect to software patents. Why not assume that the well–established answer to the definitional question in the context of identifying § 112(f) corresponding structure in the specification during claim construction can be carried over and used to identify § 112(f) structure in the claim limitations in the threshold test?[10] The flaw in this assumption lies in the overlooked, purely relational nature of the Federal Circuit's pre–*Williamson* definition of an algorithm. This definition of an algorithm can only identify software's structure in relation to the baseline provided by particular claim limitations: an algorithm is a step–by–step procedure specifying how to perform *a function recited as a claim limitation*. During the search for corresponding structure in the specification needed to construe a § 112(f) limitation, this relational definition of software's structure may be awkward at times as a policy matter, but it at least provides a conceptually coherent doctrinal rule.[11] However, it is nonsensical to use this relational definition to identify sufficient claim structure in the context of the threshold test to determine whether § 112(f) applies in the first place. Asking whether a claim limitation recites a step–by–step procedure for specifying how to perform a claim limitation makes no sense. Given that an algorithm is defined only in relation to the tasks specified as claim limitations, one cannot identify the structure in claim limitations that is needed to avoid § 112(f) under *Williamson* using the pre–*Williamson* definition of an algorithm.

Before *Williamson*, the strong presumptions of the § 112(f) threshold test and the relational definition of an algorithm allowed the Federal Circuit

9. WMS Gaming Inc. v. Int'l Game Tech., 184 F.3d 1339, 1349 (Fed. Cir. 1999).

10. The Federal Circuit has noted that "[n]aturally, there is some analytical overlap between . . . [the] two steps" of the threshold test and the identification of corresponding structure. Apple Inc. v. Motorola Inc., 757 F.3d 1286, 1294–1304 (Fed. Cir. 2014).

11. The awkwardness inheres in the formalistic, not substantive, limit on permissible claim breadth that using a relational definition of software's structure generates. *See infra* note 92.

to suppress the true extent of the sui generis law needed to bring § 112(f) to bear on software. They allowed the Federal Circuit to maintain the appearance that the application of § 112(f) to software claims was at least close to situation normal, when it was, in fact, anything but. *Williamson*, however, will pull back the curtain and force the Federal Circuit to openly grapple with the unusual nature of software's structure. This is the *Williamson* revolution.

While the call for revolution to implement *Williamson* is clear, what form that revolution will take is not. There are several ways in which the Federal Circuit can, in theory, make do with its relational definition of an algorithm in the threshold test in a post–*Williamson* world. However, the change required to harmonize the application of § 112(f) to software and the application of § 112(f) to other technologies is a revolution that defines software's structure in a stand–alone manner and that does not incorporate the functions recited in a particular claim limitation as a baseline. A stand–alone definition of software's structure would likely require a levels–of–generality analysis. Functional descriptions of a software invention exist at many different levels or rungs on a ladder of generality. At some point on the descent of this ladder, a highly general, structureless description of a software program transforms into a granular description that refers to metaphorical, but not literal, structure. Articulating such a stand–alone definition of software's structure would unquestionably be a difficult undertaking.[12] The echoes of Learned Hand's levels–of–generality test for drawing the idea/expression dichotomy in copyright are clear, and that dichotomy is notorious for its lack of ex ante clarity even without the technological complexity of software.[13] A stand–alone definition of structure could be based on what I have elsewhere described as a line between functional limitations reciting end–user preferences (that should be subject to § 112(f)) and functional limitations reciting how those end–user preferences are achieved (that should not be subject to § 112(f)).[14] Yet regardless of its final formulation, the initial step for developing the needed definition should ideally be an interdisciplinary conversation that leverages the expertise of lawyers, computer scientists, and economists to identify the levels of generality at which a functional description of software could be

12. It is precisely the difficulty of formulating such a stand–alone definition of software that has led the Federal Circuit to give only lip service to the statutory mandate in § 112(f) for step–plus–function method claims. *See infra* notes 130–132.

13. *See* Nichols v. Universal Pictures Corp., 45 F.2d 119, 121 (2d Cir. 1930) ("Nobody has ever been able to fix that boundary, and nobody ever can.").

14. *See* Collins, *supra* note 8, at 1466–67.

deemed to be logical structure for the purpose of § 112(f) as a menu of options.[15]

The initial three parts of this Essay provide background. Part II introduces § 112(f). Part III identifies why the application of § 112(f) to software requires sui generis rules that focus on logical structure, and it presents the Federal Circuit's cases that require the disclosure of an algorithm in the specification as the corresponding structure for § 112(f) software limitations. Part IV discusses the § 112(f) threshold test in greater detail and explains how *Williamson* altered it. The final two parts address life after *Williamson*. Part V argues that the Federal Circuit's pre–*Williamson* software cases provide little guidance for courts seeking to answer the definitional "What is structure?" question as part of the threshold test and that *Williamson* therefore mandates revolutionary change in § 112(f) as it applies to software. Part VI briefly maps out several different paths that the revolution could take, focusing principally on the levels–of–generality analysis that is needed to develop a stand–alone, rather than relational, definition of software's structure. Part VII concludes.

II. THE POLICY AND DOCTRINE OF § 112(F)

During the first half of the twentieth century, the Supreme Court invalidated a number of claims that employed purely functional language because such claims granted patentees rights that were overbroad with respect to the patentees' actual contributions to progress.[16] Most famously, the Court's 1946 opinion in *Halliburton Oil Well Cementing Co. v. Walker* discussed the overbreadth that inheres in the "overhanging threat of the functional claim" at length:

> Just how many different devices there are of various kinds and characters which would serve to [fulfill the claimed function] we do not know. . . . In this age of technological development there may be many other devices beyond our present information or

15. *Id.*

16. *See, e.g.*, Gen. Elec. Co. v. Wabash Appliance Corp., 304 U.S. 364, 368–71 (1938) ("The claim uses indeterminate adjectives which describe the function of the grains to the exclusion of any structural definition, and thus falls within the condemnation of the doctrine that a patentee may not broaden his product claims by describing the product in terms of function."); Holland Furniture Co. v. Perkins Glue Co., 277 U.S. 245, 257–58 (1928) ("That the patentee may not by claiming a patent on the result or function of a machine extend his patent to devices or mechanisms not described in the patent is well understood."). The explanation of why functional claims are overbroad is more complicated than is commonly acknowledged. *See* Collins, *supra* note 8, at 1411–24.

> indeed our imagination which will perform that function and yet fit these claims.[17]

While these Supreme Court cases remain good law in the sense that a patent applicant who is the first to invent a technology that performs a function cannot claim all devices that can perform that function, Congress softened their impact by enacting what is now § 112(f) as part of the 1952 Patent Act:

> An element in a claim . . . may be expressed as a means or step for performing a specified function without the recital of structure, material, or acts in support thereof, and such claim shall be construed to cover the corresponding structure, material, or acts described in the specification and equivalents thereof.[18]

Section 112(f) sanctions functional claim limitations, which can be helpful to time–constrained patent drafters because describing an invention without using functional limitations is often a difficult undertaking. However, § 112(f) exacts a price from patent drafters who use functional limitations. It codifies an exception to the default rule of claim construction specified in *Phillips v. AWH*[19]: the permitted functional claim language refers only to devices that have the "corresponding structure" or "material" that the patentee discloses in the specification, as well as its equivalents.[20] A limitation construed under § 112(f) is usually significantly narrower than the same limitation would be if it were to be construed under the default rules of claim construction specified in *Phillips* because, under § 112(f),

17. Halliburton Oil Well Cementing Co. v. Walker, 329 U.S. 1, 12 (1946). *Halliburton* is sometimes mistakenly labeled as an indefiniteness holding rather than an overbreadth holding. *See* Collins, *supra* note 8, at 1429 n.122.

18. 35 U.S.C. § 112(f) (2012).

19. Phillips v. AWH Corp., 415 F.3d 1303, 1311–24 (Fed. Cir. 2005) (en banc) (laying out the default rules of claim construction).

20. 35 U.S.C. § 112(f). Some of the Supreme Court's reasoning in its functional–claiming cases focused narrowly on the vice of using functional claiming at the point of novelty. *See, e.g.*, *Gen. Elec.*, 304 U.S. at 371. However, Congress's response to these cases was not limited to functional limitations at the point of novelty. Section 112(f), on its face, applies to all functional limitations. There are good reasons to believe breadth at a claim's point of novelty is far more problematic than breadth at limitations describing aspects of prior art technologies. *See* Kevin Emerson Collins, *Getting into the "Spirit" of Innovative Things: Looking to Complementary and Substitute Properties to Shape Patent Protection for Improvements*, 26 BERKELEY TECH. L.J. 1217 (2011); Lemley, *supra* note 8, at 958–59. This Essay, however, brackets the issue of whether § 112(f) should have more bite at a claim's point of novelty and leaves that inquiry for another day.

claim scope hews more closely to the particular embodiments disclosed in the specification.[21]

Today, the process of claim construction under § 112(f) proceeds in three phases. The first phase is a threshold test: Is a given claim limitation the type of limitation that is subject to § 112(f), or should *Phillips* determine the meaning of the claim language? The doctrinal formulation of the threshold test has shifted over time—and was most recently altered by *Williamson*—but, to one degree or another, it has always involved a search for sufficient structure in the claim limitation.[22] The more structure that a claim limitation recites, the less pressing the policy concerns about overbroad, functional claiming and the less likely the scope–limiting rule of § 112(f) is to govern claim construction. The second phase employs the default rules of claim construction to interpret the meaning of the functional language employed in the claim.[23] The third phase then identifies the "corresponding structure" disclosed in the specification that is capable of performing the disclosed function.[24] In order to fall within literal claim scope, a technology must literally perform the function specified in the claim limitation, and its structure must be the corresponding structure disclosed in the specification or its equivalents.

While the first and third phases both involve a search for structure, each one looks for structure in a different place in the patent document. The first phase (the threshold test) looks for structure in the claim limitations

21. Articulating an interesting historical argument, John Duffy argues that prior to the creation of the Federal Circuit, the default rule of claim construction for all limitations resembled the rule articulated in § 112(f) and that the stakes of the § 112(f) threshold test were therefore low. John F. Duffy, *Counterproductive Notice in Literalistic Versus Peripheral Claiming*, 96 B.U. L. REV. 1197, 1206–10 (2016). Professor Duffy also argues as a statutory matter that Congress intended § 112(f) to bring the default rule of claim construction to bear on functional claims. *Id.* According to Professor Duffy, it was the Federal Circuit's shift to a broader, "literalistic" rule of claim construction by the 1990s that "transformed" § 112(f) into a scope–narrowing rule of claim construction and that introduced economic importance into the § 112(f) threshold test. *Id.*

22. The threshold test and *Williamson*'s impact thereon are addressed at greater length below. *See infra* Part IV.

23. *See Generation II Orthotics, Inc. v. Med. Tech., Inc.,* 263 F.3d 1356, 1364–65 (Fed. Cir. 2001).

24. That is, the specification must contain descriptive text by which a person of skill in the field of the invention would "know and understand what structure corresponds to the means limitation." Finisar Corp. v. DirecTV Grp., Inc., 523 F.3d 1323, 1340 (Fed. Cir. 2008). An enabling specification is not enough. Biomedino LLC v. Waters Techs. Corp., 490 F.3d 946, 953 (Fed. Cir. 2007). The disclosed structure must also be clearly linked to the function recited in the claim limitation. B. Braun Med., Inc. v. Abbott Labs., 124 F.3d 1419, 1424 (Fed. Cir. 1997).

themselves to determine whether § 112(f) governs. In contrast, the third phase looks for structure in the specification in order to determine the scope of a claim limitation that is governed by § 112(f).

If a limitation is governed by § 112(f) and the specification does not disclose any corresponding structure for that limitation, then the claim is invalid for indefiniteness under § 112(b) of the Patent Act.[25] Indefiniteness holds that claims that employ limitations whose meanings cannot be ascertained with reasonable certainty are invalid.[26] Indefiniteness is a common–sense rule: the scope of a claim to a "thingamajig" cannot be ascertained, there is poor public notice, and there are many instances in which neither the validity nor the infringement analyses can proceed. Section 112(f) states that a functional claim limitation refers to the corresponding structure and its equivalents, so a § 112(f) limitation has no discernable meaning if the specification does not disclose any corresponding structure.[27]

The apparatus limitations that are governed by § 112(f) are frequently called "means–plus–function" limitations. However, § 112(f) also governs action limitations in method claims. A claim limitation "may be expressed as a . . . step for performing a specified function without the recital of . . . acts in support thereof, and such claim shall be construed to cover the corresponding . . . acts described in the specification and equivalents thereof."[28] Following the statue, a "step" is a generic element of a process, and an "act" refers to one of the sub–steps that are needed to implement a "step."[29] While the Federal Circuit has occasionally discussed the notion of a step–plus–function method claim that is subject to § 112(f), it has never applied § 112(f) to an action in a method claim.[30]

25. *In re* Donaldson Co., 16 F.3d 1189, 1195 (Fed. Cir. 1994) (en banc).

26. Nautilus, Inc. v. Biosig Instruments, Inc., 134 S. Ct. 2120, 2124 (2014). Indefiniteness derives from the statutory requirement that a patent "particularly point[] out and distinctly claim[] the subject matter which the inventor regards as the invention." 35 U.S.C. § 112(b) (2012).

27. *Donaldson*, 16 F.3d at 1195.

28. 35 U.S.C. § 112(f) (2012).

29. O.I. Corp. v. Tekmar Co. Inc., 115 F.3d 1576, 1582–83 (Fed. Cir. 1997).

30. *See, e.g.*, Masco Corp. v. United States, 303 F.3d 1316, 1326–28 (Fed. Cir. 2002); Seal-Flex, Inc. v. Athletic Track & Court Const., 172 F.3d 836, 848–51 (Fed. Cir. 1999) (Rader, J., concurring); *O.I. Corp.*, 115 F.3d at 1582–84.

III. SECTION 112(F) AND SOFTWARE

The application of § 112(f) to software requires a sui generis modification of conventional § 112(f) doctrine. Section III.A posits that this modification is required because software is, for practical purposes at least, a purely functional technology without any physical structure that is relevant to what a software inventor invents. Section III.B explores how the Federal Circuit had already adapted § 112(f) to software before *Williamson* by concluding that an algorithm is the corresponding structure in the specification for software means–plus–function limitations.

A. THE SOFTWARE/§ 112(F) MISMATCH[31]

There is a fundamental mismatch between the rules of § 112(f) and software inventions: in most technologies, § 112(f) builds on an intuitive distinction between the physical structure and the function of a technology, but in software, § 112(f) cannot build on this distinction. An economically rational patent regime cannot require that the physical, structural properties of a software program be recited as claim limitations. Software inventions are, at least as a practical matter and for the purpose of patent law, a purely functional technology.

In the historical core of § 112(f) claiming in the mechanical arts, the principles distinguishing the structural properties of an invention from its functional properties are self–evident, intuitive concepts. In brief, a structural property is what an invention "is," whereas a functional property is what an invention "does."[32] For example, a "skid plate,"[33] "button and hole arrangement,"[34] and "a rotatable disc with an opening sits above the

31. The argument in this Section draws from Collins, *supra* note 8, at 1440–43.

32. *In re* Swinehart, 439 F.2d 210, 212 (C.C.P.A. 1971). The distinction between structural and functional properties is sufficiently basic and intuitive that it often underlies the philosophical analysis of the properties of things themselves. *See, e.g.*, Peter Kroes, *Technological Explanations: The Relation between Structure and Function of Technological Objects*, 3 SOC'Y FOR PHIL. & TECH. 18, 18 (1998) (discussing "two different modes of description, viz., a *structural* and a *functional* mode of description" for technological objects).

33. Chiuminatta Concrete Concepts, Inc. v. Cardinal Indus., Inc., 145 F.3d 1303, 1309 (Fed. Cir. 1998) (noting that the specification described a skid plate as "a generally rectangular strip of metal having rounded ends . . . between which is a flat piece" where "[t]he flat piece . . . is generally parallel to the base plate").

34. Al-Site Corp. v. VSI Int'l, Inc., 174 F.3d 1308, 1315–16 (Fed. Cir. 1999).

container cap"[35] are all structural descriptions of objects.[36] Few—if any—cases in the mechanical arts bother even to expressly pose the definitional "What is structure?" question because the answer is so self–evident. Of course, this clarity does not mean that the outcome of the § 112(f) threshold test is never contested in the mechanical arts. There are many cases in the mechanical arts in which it is unclear whether the quantum of structure recited as a claim limitation is sufficient to avoid § 112(f).[37] However, this uncertainty inheres in either the meaning of language or the answer to the quantitative question of the threshold test, not the answer to the definitional question of the threshold test. That is, the uncertainty follows from whether claim language refers to enough of the structural properties of a technology, not whether any particular property, once identified as a claim limitation, is a structural property.

In the software arts, however, § 112(f) cannot leverage the intuitive distinction between physical structure and function into a convenient proxy for limiting permissible claim scope. Of course, software programs do have physical, structural properties, just like other technologies do. Software may be commonly described as intangible, [38] but this description is technically inaccurate.[39] Software exists as electrons or charges on a hard drive or in a computer's memory; a computer implements a software program only because a particular set of gates or switches in the processor is open or

35. Sage Prods., Inc. v. Devon Indus., Inc., 126 F.3d 1420, 1428 (Fed. Cir. 1997). Here, "rotatable" adds a dollop of function to the description.

36. Structure is also an uncontroversial concept in the chemical and biochemical arts because structure is equated with molecular structure. *See* Ariad Pharm., Inc. v. Eli Lilly & Co., 598 F.3d 1336, 1350 (Fed. Cir. 2010) (en banc) (treating the recitation of molecular structure in a claim limitation as the type of structure that supports the claim's validity). However, the task of policing overbroad, functional claims in these arts has fallen to the written description doctrine rather than to means–plus–function claiming. Collins, *supra* note 8, at 1430–33.

37. *Cf. supra* note 7 and accompanying text (discussing the quantitative question of the § 112(f) threshold test).

38. *See, e.g.*, Bancorp Servs. v. Sun Life Assurance Co., 687 F.3d 1266, 1277–79 (Fed. Cir. 2012) (making an analogy between computer-executed and mental processes); *In re* Grams, 888 F.2d 835, 840 (Fed. Cir. 1989) (labeling software–executed processes as non–physical steps); Richard S. Gruner, *Intangible Inventions: Patentable Subject Matter for an Information Age*, 35 LOY. L.A. L. REV. 355, 357 (2002) ("New designs for software and computer-based business practices . . . resemble the sorts of intangible ideas and thought processes that have traditionally fallen outside of patent protections.").

39. Microsoft Corp. v. AT&T Corp., 550 U.S. 437 (2007) (discussing the physicality of any copy of a software program that can generate functional effects in the course of assessing when software can be a "component" under § 271(f)).

closed.[40] These are software's physical, structural properties. The crux of the problem for § 112(f) is thus not that software lacks physical, structural properties but rather that those properties are usually irrelevant to the task of identifying, delineating, or defining a protectable software invention. This irrelevance can be seen on two distinct levels. First, minor changes in code can lead to significant changes in software's physical, structural properties. A software invention can be implemented in entirely different code in the same programming language or in an entirely different language, and the sets of code that all embody the invention have few, if any, structural properties in common.[41] Second, executing the same code on different hardware leads to radical changes in software's physical, structural properties. Thanks to interpreters and compilers, any given program can be implemented on a wide array of different computers, each possessing a different internal architecture and each requiring the software to adopt entirely different physical, structural properties to yield the intended, functional result.[42] In sum, the many distinct embodiments of a software invention are unlikely to have any physical, structural properties in common that can be used to delineate what an inventor has invented. Any attempt to define software by its physical, structural properties will fail.[43]

The irrelevance of software's physical, structural properties to the definition of a software invention means that, as a practical matter, software is a purely functional technology. In the software arts, an invention is its function, not its structure.[44] For patent drafters writing software claims,

40. WMS Gaming Inc. v. Int'l Game Tech., 184 F.3d 1339, 1348 & n.3 (Fed. Cir. 1999); Robert Plotkin, *Computer Programming and the Automation of Invention: A Case for Software Patent Reform*, 7 UCLA J.L. & TECH. 1, 38–39 (2003) ("[T]he physical structure of the computer is critical if software is to execute and thereby perform its intended functions."); *cf. In re* Alappat, 33 F.3d 1526, 1545 (Fed. Cir. 1994) (en banc) (holding that a computer programmed with a new software program is a new machine under the novelty doctrine that is structurally distinct from prior art machines).

41. Furthermore, hardware and software implementations of any given program are functionally interchangeable despite their radically different structural properties. *In re* Alappat, 33 F.3d 1526, 1583 (Fed. Cir. 1994) (Rader, J., concurring).

42. *See* W. DANIEL HILLIS, THE PATTERN ON THE STONE 56–58 (1998) (discussing interpreters and compilers); *cf. Microsoft Corp.*, 550 U.S. at 450 ("Software . . . is a stand-alone product developed and marketed for use on many different types of computer hardware." (internal quotations omitted)).

43. Plotkin, *supra* note 40, at 46 & n.126 ("For all practical purposes the programmer and others who think about and describe the program have no practical choice but to conceive of and describe it in terms of its logical structure [or function]. . . . It is far from clear that it would even be possible for the human mind to appreciate the physical structure of all but the simplest programs or to explain them in terms of their physical structures.").

44. Note, *Everlasting Software*, 125 HARV. L. REV. 1454, 1456 (2012).

software functionality is like a never–ending set of nested Russian dolls: you open up one more general functional description to look for structure, and all you find is another, more specific functional description.[45] The purely functional nature of a software program is not an accident. To the contrary, it is a key feature that has been engineered into the very nature of software. The value of modern software lies in the fact that the coding of a software program that possesses any given set of logical, functional properties need not involve any consideration of software's physical, structural properties.[46]

The purely functional nature of software raises a challenge for any patent doctrine that builds on the categorical distinction between the structural and functional properties of an invention and prohibits purely functional claim limitations. This mismatch between software and § 112(f) presents two doctrinally simple solutions, but each is extreme and unpalatable.[47] On the one hand, we could refuse to make any allowances for the purely functional nature of software and mandate that software patents recite physical, structural properties as limitations on claim scope. This option would render patent protection for software absurd and economically meaningless because avoiding infringement would be a trivial undertaking. On the other hand, we could exempt software from the strictures of § 112(f) and permit purely functional claiming at any level of generality. This option ignores the Supreme Court's cases from the early twentieth century that prohibit purely functional claiming and raises significant policy concerns about the social costs of software–claim overbreadth.[48] Given the problematic nature of both simple, but extreme, doctrinal solutions, an exploration of the sui generis ways in which § 112(f) could be modified for a purely functional art like software is a worthwhile undertaking. Exploring

45. Technically, the nested dolls do end at some point. *See infra* note 57 (discussing *In re* Katz).

46. Plotkin, *supra* note 40, at 36 ("[O]ne of computer science's express goals is to ensure that programmers can do their work in complete ignorance of the physical structure of . . . hardware."); *see also id.* at 26 ("The process of computer programming enables a programmer to create a machine that has a particular novel physical structure for performing a particular function without requiring the programmer to design the novel features of the machine in physical terms."); *id.* at 44–45 ("[A] programmer who modifies the physical structure of a computer by providing source code to the computer need not even know that the computer's memory is being physically modified at all, much less understand or appreciate the nature of those physical modifications.").

47. If either were adopted, it might well be preferable to eliminate patent protection for software altogether.

48. *See supra* notes 16–17 and accompanying text.

when function should count as metaphorical structure offers the greatest promise for a way to use § 112(f) to navigate a course between the Scylla and Charybdis of trivially narrow and vastly overbroad software claims.

B. ALGORITHMS AS CORRESPONDING STRUCTURE

The need to create a sui generis doctrine for answering the definitional, "What is structure?" question of § 112(f) for software claims has not been lost on the Federal Circuit. The Federal Circuit has developed an extensive body of case law that identifies an algorithm as the corresponding structure in a patent disclosure for a § 112(f) software limitation.[49]

The Federal Circuit first introduced the notion of an algorithm as the corresponding structure in its 1999 opinion in *WMS Gaming, Inc. v. International Game Technologies*.[50] The district court construed a functional software limitation employing the term "means" broadly to encompass any table, formula, or algorithm for performing the claimed function, but the Federal Circuit insisted that the scope of the claim was limited to "the special purpose computer programmed to perform the disclosed algorithm" in the specification, as well as its equivalents.[51] In subsequent cases, the Federal Circuit has expressly defined an algorithm as "a series of instructions for the computer to follow,"[52] "a step-by-step procedure for accomplishing a given result,"[53] and a "sequence of computational steps to follow."[54] Simply put, "the essence of algorithms" is "what to do to perform a task" that is recited as a claim limitation.[55]

One line of § 112(f) cases in particular offers a good window through which to examine what constitutes an algorithm under the Federal Circuit's

---

49. *See infra* notes 50–59 and accompanying text. However, the Federal Circuit has not developed a definition of software's structure that can function in the § 112(f) threshold test. *See infra* Part V.

50. WMS Gaming Inc. v. Int'l Game Tech., 184 F.3d 1339, 1347–50 (Fed. Cir. 1999).

51. *Id.* at 1349.

52. Typhoon Touch Techs., Inc. v. Dell, Inc., 659 F.3d 1376, 1384 (Fed. Cir. 2011) (quoting *In re* Waldbaum, 457 F.2d 997, 998 (C.C.P.A. 1972)).

53. *Id.* at 1385 (quoting *In re* Freeman, 573 F.2d 1237, 1254 (C.C.P.A. 1978)).

54. Ibormeith IP, LLC v. Mercedes-Benz USA, LLC, 732 F.3d 1376, 1379 (Fed. Cir. 2013) (citations omitted). The Federal Circuit has also stated that there is no single format in which an algorithm must be communicated. Mathematical formulae, prose, and flow charts can all express algorithms. Finisar Corp. v. DirecTV Grp., Inc., 523 F.3d 1323, 1340 (Fed. Cir. 2008).

55. Allen Newell, *The Models Are Broken, The Models Are Broken!*, 47 U. PITT. L. REV. 1023, 1026 (1986); *see also id*. at 1024 ("An algorithm is [a] . . . sequence of steps or operations for solving a class of problems.").

definition. Starting in the late 2000s, the Federal Circuit began to invalidate a large number of software claims with § 112(f) limitations for indefiniteness because the specification did not disclose an algorithm (and thus it did not disclose corresponding structure).[56] If the specification simply repeats the functions recited in the claims, the specification does not disclose an algorithm. For example, in *Finisar Corp. v. Direct TV Group, Inc.*, the Federal Circuit held a § 112(f) software limitation to be indefinite because the specification provided "nothing more than a restatement of the function, as recited in the claim."[57] However, if the specification provides a more granular functional description of the software than the claim does—that is, if it discloses a series of functional steps that, when strung together, achieve the claimed function—then the specification discloses an algorithm. For example, in *Typhoon Touch Techs. v. Dell*, a claim recited the § 112(f) limitation "means for cross-referencing said responses with one

56. *See* Eon Corp. IP Holdings LLC v. AT&T Mobility LLC, 785 F.3d 616, 621–24 (Fed. Cir. 2015); Triton Tech, LLC v. Nintendo, Inc., 753 F.3d 1375, 1378–80 (Fed. Cir. 2014); Robert Bosch, LLC v. Snap-On Inc., 769 F.3d 1094, 1011–12 (Fed. Cir. 2014); Function Media, L.L.C. v. Google Inc., 708 F.3d 1310, 1317–19 (Fed. Cir. 2013); Dealertrack, Inc. v. Huber, 674 F.3d 1315, 1330 (Fed. Cir. 2012); ePlus, Inc. v. Lawson Software, Inc., 700 F.3d 509, 518–20 (Fed. Cir. 2012); Noah Sys., Inc., v. Intuit Inc., 675 F.3d 1302, 1311–19 (Fed. Cir. 2012); *In re* Aoyama, 656 F.3d 1293, 1296–1300 (Fed. Cir. 2011); Rembrandt Data Techs. v. AOL, 641 F.3d 1331, 1338–43 (Fed. Cir. 2011); *In re* Katz Interactive Call Processing Patent Litig., 639 F.3d 1303 (Fed. Cir. 2011); Brown v. Baylor Healthcare Sys., No. 2009-1530, 2010 WL 1838921, at *3–*4 (Fed. Cir. May 7, 2010); Net MoneyIN, Inc. v. VeriSign, Inc., 545 F.3d 1359, 1366–67 (Fed. Cir. 2009); Encyc. Britannica, Inc. v. Alpine Elecs., Inc., No. 2009-1087, 2009 WL 4458527, at *3–*6 (Fed. Cir. Dec. 4, 2009); Blackboard, Inc. v. Desire2Learn Inc., 574 F.3d 1371, 1382–85 (Fed. Cir. 2008); Finisar Corp. v. DirecTV Grp., Inc., 523 F.3d 1323, 1339–41 (Fed. Cir. 2008); Aristocrat Techs. Austl. Pty. Ltd. v. Int'l Game Tech., 521 F.3d 1328, 1332–38 (Fed. Cir. 2008).

57. *Finisar Corp.*, 523 F.3d at 1340. The Federal Circuit has identified one category of functional limitations in software claims that do not require the disclosure of any corresponding structure. In *Katz Interactive Call Processing Patent Litig. v. Am. Airlines, Inc*., the Federal Circuit held that the claimed functions of "processing," "receiving," and "storing" did not require the disclosure of an algorithm even though the limitations were subject to § 112(f) because such functions "can be achieved by any general purpose computer without special programming" and thus "it was not necessary to disclose more structure than the general purpose processor that performs those functions." 639 F.3d 1303, 1316 (Fed. Cir. 2011). Subsequent Federal Circuit cases that have considered the *Katz* exception have called it "narrow," found it not to apply, and required the disclosure of an algorithm in the specification to avoid indefiniteness. *See EON Corp.*, 785 F.3d at 621–23 (Fed. Cir. 2015); Ergo Licensing, LLC v. CareFusion 303, Inc., 673 F.3d 1361, 1364 (Fed. Cir. 2012).

of said libraries of said possible responses."[58] The Federal Circuit upheld the claim as definite because the specification "recite[d] a four-step algorithm for computer-implemented cross-referencing, starting with the entry of a response, then a search for the entered response in a library of responses, then determination whether a match exists in the library, and then execution of an action if a match exists."[59] The specification described a series of four functional tasks, each of which was more granular than the functional task recited as a claim limitation. When strung together, these disclosed tasks formed an algorithm for accomplishing the claimed task.

Together, the adoption of an algorithm as software's corresponding structure in *WMS Gaming* and the subsequent cases addressing the indefiniteness of software claims with § 112(f) limitations that fail to disclose an algorithm demonstrate that the Federal Circuit has adopted a sui generis definition of structure in the software arts. In most arts, the Federal Circuit defines structure for the purposes of § 112(f) in terms of a technology's physical, spatial, and material properties,[60] but, in software, it rejects equating software's structure with these physical properties on two different levels. First, the Federal Circuit rejects the notion that the disclosure of a generic computer or processor in the specification amounts to the disclosure of corresponding structure.[61] However, standing alone, this rejection could be taken to mean that a generic computer or processor is not *enough* physical structure.[62] It leaves open the possibility that the required structure is the physical structure that distinguishes one software program from another, and the disclosure of a generic computer cannot mark this distinction.[63] Second, by embracing an algorithm as corresponding structure, the Federal Circuit's § 112(f) software cases implicitly reject physical structure altogether. An algorithm is not the same type of structure

58. Typhoon Touch Techs., Inc. v. Dell, Inc., 659 F.3d 1376, 1383 (Fed. Cir. 2011) (citation omitted).

59. *Id.* at 1385.

60. *See supra* notes 32–36.

61. *See, e.g.*, *EON Corp.*, 785 F.3d at 621; *Aristocrat Techs.*, 521 F.3d at 1333; WMS Gaming Inc. v. Int'l Game Tech., 184 F.3d 1339, 1349 (Fed. Cir. 1999).

62. The analogy in the mechanical arts to the disclosure of a generic computer as corresponding structure might be the disclosure of a material like metal as the corresponding structure. Being made of metal is a structural property of a mechanical invention, but it does not provide enough information about structure to say that the specification has disclosed the corresponding structure.

63. *Cf. supra* note 40 and accompanying text (proposing that software's most relevant physical structure is a distribution of electronics on a storage medium or an arrangement of open and closed gates in a computer).

considered in § 112(f) for other technologies. Whereas the structure of other technologies for purposes of § 112(f) is physical structure, the steps that make up an algorithm are still functional in a literal sense. For example, each of the steps in the *Typhoon Touch Technologies* algorithm—such as the step of "search[ing] for the entered response in a library of responses"—is just as functional as the claimed task—namely "cross-referencing said responses with one of said libraries of said possible responses."[64] The difference between the claimed function and the functions disclosed in the patent specification is only the level of granularity of the functional description: the disclosed algorithm is still a functional description of what the software does, just one that is more specific than the functional description provided in the claims.[65] This definition of software's structure not as physical structure but rather as logical or metaphorical structure makes software a sui generis technology under § 112(f).[66] In no technology other than software does a sufficiently specific functional description of a technology ever get counted as corresponding structure.[67]

---

64. *See supra* notes 58–59.

65. The language that the Federal Circuit employs to describe what it is doing, however, does not always overtly reflect its practice. The Federal Circuit occasionally juxtaposes the language expressing an algorithm with functional language, implicitly supporting the (incorrect) inference that an algorithm is something other than a functional entity. For example, in order to explain its invalidation of a software claim with a § 112(f) limitation for indefiniteness, the Federal Circuit stated in *Ergo Licensing v. CareFusion* that "[t]he specification merely provides functional language and does not contain any step-by-step process for controlling the adjusting means." 673 F.3d 1361, 1365 (Fed. Cir. 2012). By implying that an algorithm—i.e., the step–by–step process—is not a form of functional language, the Federal Circuit ignores its own definition of an algorithm and sweeps the sui generis nature of software under the rug. To avoid an outright contradiction, the Federal Circuit's language in passages like this one must be interpreted to mean that the specification cannot merely provide functional language at the same level of generality as the functional language in the claim limitation.

66. The adoption of metaphorical structure in the software arts means that we are talking about different ideas altogether when we discuss structure for the purpose of § 112(f) and structure for the purpose of the § 101 prohibition on claims to propagating signals or software per se, articulated in *In re Nuijten*, 500 F.3d 1346, 1356 (Fed. Cir. 2007) ("While such a signal is physical and real, it does not possess concrete structure."); *id.* at n.6 (noting that the signal on a "storage medium" would be patent eligible). The structure required by *In re Nuijten* refers to physical structure, but the structure needed to satisfy § 112(f) does not.

67. Thus, "'[s]tructure' to a person of ordinary skill in the art of computer-implemented inventions may differ from more traditional, mechanical structure." Apple Inc. v. Motorola Inc., 757 F.3d 1286, 1298 (Fed. Cir. 2014). In this same passage, the Federal Circuit also states that "looking for traditional 'physical structure' in a computer software claim is fruitless because software does not contain physical structures." *Id*. This second statement is technically inaccurate. Software does not violate materialism; it does

IV. *WILLIAMSON* AND THE § 112(F) THRESHOLD TEST

The § 112(f) threshold test determines whether a claim limitation is subject to the default rule of claim construction articulated in *Phillips* or the scope–narrowing rule of claim construction specified in § 112(f). Since the middle of the 1990s, two competing theories of the reach of § 112(f) have struggled for dominance in the Federal Circuit: *intent theory* and *scope theory*. Neither has ever completely dominated the other, but the weight given to each has varied over time. For at least the decade prior to *Williamson*, intent theory had the upper hand. With *Williamson*, the Federal Circuit gave the upper hand to scope theory.

Intent theory suggests that § 112(f) is an option that Congress placed at the disposal of patent drafters and that § 112(f) should apply when, and only when, patent drafters use the word "means" as a signal that they intend to invoke it. Drawing on the literal text of the statute, intent theory posits that § 112(f):

> provides that an element in a claim for a combination 'may be expressed' as a means for performing a function, which indicates that the patentee is afforded the option of using the means-plus-function format. The question then is whether, in the selection of claim language, the patentee must be taken to have exercised that option.[68]

Thus, intent theory gives rise to twin presumptions in the threshold test: "the term 'means' (particularly as used in the phrase 'means for') generally invokes § [112(f)] and . . . the use of a different formulation generally does not."[69] At least on paper, these presumptions have always been rebuttable: enough structure accompanying the word "means" could take the limitation out of the ambit of § 112(f), and insufficient structure accompanying a non–means term could lead to the application of § 112(f).[70]

In contrast, scope theory dismisses the formalism of semantic word choice and focuses directly on whether the claim limitation recites enough structure so that the Supreme Court's policy concerns about overbroad

---

have physical structure. *See supra* note 40 and accompanying text. However, the Federal Circuit's statement captures the spirit of the law insofar as it implies that physical structure is and should be irrelevant in the § 112(f) analysis of software claims.

68. Greenberg v. Ethicon Endo-Surgery, Inc., 91 F.3d 1580, 1584 (Fed. Cir. 1996); *see also* York Prods., Inc. v. Cent. Tractor Farm & Family Ctr., 99 F.3d 1568, 1574 (Fed. Cir. 1996).

69. *Greenberg*, 91 F.3d at 1584.

70. *York Prods.*, 99 F.3d at 1574.

claim scope raised in *Halliburton Oil Well Cementing* are mitigated or eliminated.[71] That is, under scope theory, a claim limitation should be subject to § 112(f) whenever it fails to "recite a definite structure which performs the described function," regardless of the precise words employed.[72] Here, § 112(f) is mandatory, not optional, whenever a claim is drafted with excessively functional language.

By the 2000s, intent theory gained the upper hand in the Federal Circuit. The presumption against the application of § 112(f) to a limitation that did not use the term "means" evolved into a "strong" presumption[73] that could only be overcome by "a showing that the limitation essentially is devoid of anything that can be construed as structure."[74] The § 112(f) threshold test thus became highly formalistic, and the scope–limiting effect of § 112(f) could easily be avoided by any knowledgeable, motivated patent drafter.

In 2014, the conflict between intent and scope theories resurfaced in the disagreement between the Federal Circuit's majority and dissenting opinions in *Apple v. Motorola* over whether a claim limitation not using the term "means" was subject to § 112(f).[75] The majority adopted a strong version of intent theory. "The strong presumption created by not including means in a claim limitation . . . signals to the court that the patentee has chosen to . . . avoid[] the benefits of Section 112, ¶ 6."[76] In contrast, the

71. *See supra* notes 16–17 and accompanying text (discussing *Halliburton*'s invalidation of functional claims as overbroad).

72. Cole v. Kimberly-Clark Corp., 102 F.3d 524, 531 (Fed. Cir. 1996). Scope theory may have been quite novel in the § 112(f) threshold test in the 1990s. *See, e.g.*, Mas-Hamilton Grp. v. LaGard, Inc., 156 F.3d 1206, 1213–14 (Fed. Cir. 1998) (holding for the first time in the Federal Circuit that a phrase not using "means" was governed by § 112(f)); *cf.* Lighting World, Inc. v. Birchwood Lighting, Inc., 382 F.3d 1354, 1359–60 (Fed. Cir. 2004) (labeling *Mas-Hamilton* as an "exceptional case").

73. *Lighting World,* 382 F.3d at 1358. In fact, intent theory had become so strong that a claim limitation without the word "means" was not a § 112(f) limitation even if it was purely functional in the sense that it encompassed all structures that were capable of performing the claimed function. *Id.* at 1361–62.

74. Flo Healthcare Sols., LLC v. Kappos, 697 F.3d 1367, 1374 (Fed. Cir. 2012). Even under the strong–presumption formulation of the threshold test, the Federal Circuit's opinions are laced with scope–theory sound bites. *Lighting World*, 382 F.3d at 1360 (noting that § 112(f) does not apply to a term "that is simply a nonce word or a verbal construct that is not recognized as the name of structure and is simply a substitute for the term 'means for'"). However, very few claim limitations that did not employ the word "means" were held to actually be governed by § 112(f). *But see* Welker Bearing Co. v. PHD, Inc., 550 F.3d 1090, 1096–97 (Fed. Cir. 2008); MIT v. Abacus Software, 462 F.3d 1344, 1355 (Fed. Cir. 2006); *Mas-Hamilton*, 156 F.3d at 1213–14.

75. Apple Inc. v. Motorola, Inc., 757 F.3d 1286, 1297 (Fed. Cir. 2014).

76. *Id.*

dissent argued that, under the majority opinion, "one minor drafting decision greatly expands the scope of the claim limitation" and "patent applicants are able to claim broad functionality without being subject to the restraints imposed by § 112 ¶ 6."[77]

Only a year after *Apple v. Motorola*, the Federal Circuit acted en banc in *Williamson v. Citrix Online* to undermine the primacy of intent theory and elevate scope theory in the threshold test.[78] Drawing from the policy concerns about the overbreadth of functional claims that led the Supreme Court to invalidate functional claims in the first place, the Federal Circuit noted that a strong presumption against using § 112(f) when a claim limitation did not use the word "means":

> has the inappropriate practical effect of placing a thumb on what should otherwise be a balanced analytical scale. It has shifted the balance struck by Congress in passing § 112, para. 6 and has resulted in a proliferation of functional claiming untethered to § 112, para. 6 and free of the strictures set forth in the statute.[79]

The Federal Circuit proceeded to flatly state that the "heightened burden" for overcoming the presumption against the application of § 112(f) in the absence of the term means "is unjustified."[80] To articulate the new threshold test, the Federal Circuit reverted to scope theory, holding that the presumption against the application of § 112(f) in the absence of the word "means" could be overcome whenever "the words of the claim are understood by persons of ordinary skill in the art to have a sufficiently definite meaning as the name for structure."[81]

---

77. *Id.* at 1337 (Prost, J., concurring in part and dissenting in part).

78. Williamson v. Citrix Online, LLC, 792 F.3d 1339 (Fed. Cir. 2015) (en banc). Only the section addressing the proper standard for the § 112(f) threshold was authored en banc.

79. *Id.* at 1349.

80. *Id. But see id.* at 1358 (Newman, J., dissenting) ("[I]t is the applicant's choice during prosecution whether or not to invoke paragraph 6, and the court's job is to hold the patentee to his or her choice.").

81. *Id.* at 1349. Presumably, the default claim construction methodology for non–112(f) limitations outlined in *Phillips v. AWH*, 415 F.3d 1303 (Fed. Cir. 2005) (en banc), should govern the search for sufficient structure in the § 112(f) threshold test. *See, e.g.*, Personalized Media Commc'ns, LLC v. U.S. Int'l Trade Comm'n, 161 F.3d 696, 704 (Fed. Cir. 1998) (stating that, in the § 112(f) threshold test, "the focus remains on whether the claim as properly construed recites sufficiently definite structure to avoid the ambit of § 112, ¶ 6"). The use of *Phillips* here makes sense as a policy matter. The Supreme Court's concerns about overbroad functional claims are only mitigated when the scope of the claim in infringement proceedings has sufficient structural limitations, and *Phillips* controls claim scope for infringement purposes. However, the Federal Circuit often speaks of the

However, *Williamson* does not represent a complete triumph of scope theory over intent theory. The Federal Circuit did not abandon the use of presumptions in the § 112(f) threshold test. The presumption that § 112(f) does not apply to limitations without the term "means" endures, and *Williamson* only makes it easier to rebut.[82] The converse, pre–*Williamson* strong presumption that the use of the word "means" triggers § 112(f) remains in place.[83]

V. THE NEED FOR A REVOLUTION

At first glance, *Williamson* might not appear to call for a revolution in the § 112(f) threshold test for software claims. While *Williamson*'s turn to scope theory will force the Federal Circuit to ask and answer the definitional "What is structure?" question as part of the threshold test on a more frequent and open basis, the Federal Circuit has already discussed software's structure in § 112(f) cases in two distinct doctrinal contexts in its pre–*Williamson* opinions. First, it has entertained arguments about the presence or absence of structure in claim limitations that were intended to rebut the

search for sufficient structure in the threshold test not in the exact terms outlined in *Phillips*, but rather in different terms that mirror *Phillips* only in a rough manner. For example, Federal Circuit cases do not usually ask whether a claim limitation has a "meaning" that is sufficiently structural, which is the usual way of talking about claim construction, but rather whether a limitation "connotes" sufficient structure. *See, e.g.*, Apex Inc. v. Raritan Comput., Inc., 325 F.3d 1364, 1373 (Fed. Cir. 2003) ("The threshold issue for all the limitations involving the term 'circuit' is whether the term itself connotes sufficient structure to one of ordinary skill in the art to perform the functions identified by each limitation."). A connotation standard may identify sufficient structure to avoid § 112(f) even when that structure does not amount to a claim limitation under *Phillips*. In some cases, the Federal Circuit seems far more willing to read structure from the specification into the claims as part of the search for sufficient structure in the threshold test than it would be to read limiting features of preferred embodiments into the claim limitations under *Phillips*. *See, e.g.*, *infra* note 113 and accompanying text (discussing *Apple*, 757 F.3d at 1300). This willingness to rely on the specification to identify sufficient structure in a claim limitation may have resulted, in part, from the now–defunct strong presumption against the invocation of § 112(f) when the limitation does not employ the term "means." It is unclear whether this willingness will persist after *Williamson*.

82. *Williamson*, 792 F.3d at 1349. A *Williamson* concurrence notes that the majority does not clarify what force the not–strong presumption against § 112(f) that follows from the absence of "means" has after *Williamson*. *Id.* at 1357 n.2 (Reyna, J., concurring in part and dissenting in part). Earlier Federal Circuit opinions suggest that the burden of persuasion is a preponderance of the evidence. *See, e.g.*, *Apex*, 325 F.3d at 1372. But, whether *Williamson* returns to this burden is unclear.

83. *Williamson,* 792 F.3d at 1349.

pre–*Williamson*, strong presumptions of the threshold test.[84] Second, after limitations have been labeled as § 112(f) limitations, it has looked for corresponding structure in the specification in the form of algorithms. However, upon closer examination, neither type of pre–*Williamson* case provides much guidance for courts trying to answer the definitional question about software's structure as part of the threshold test. Starting with the better–developed body of law, Section V.A demonstrates that the Federal Circuit's existing algorithm jurisprudence cannot identify structure as part of the threshold test because an algorithm is a relational entity that can only be discerned by comparing a functional description of how a software program works to a claim limitation. Section V.B then looks for crumbs of wisdom in the Federal Circuit's pre–*Williamson* cases that address the definitional question as part of the strong–presumption threshold test.

A. PRECEDENT ON ALGORITHMS AS CORRESPONDING STRUCTURE

There are two distinct problems with using the Federal Circuit's definition of an algorithm that was developed to identify corresponding structure in the specification during claim construction in order to identify sufficient structure in the claims as part of the § 112(f) threshold test. First, § 112(f) has traditionally required each limitation, considered independently, to recite sufficient structure, but no single, functionally defined claim limitation can ever be an algorithm. Second, even if the § 112(f) threshold test is altered so that it looks for an algorithm in a series of functional claim limitations, the inherently relative nature of the Federal Circuit's definition of an algorithm leads to an unworkable conceptual muddle.

No single, functional claim limitation can ever be an algorithm because identifying a single claim limitation as an algorithm involves a category error. Functional claim limitations are singular, and algorithms are plural. As the Federal Circuit has defined the concept, an algorithm is a sequence of steps for performing a task.[85] Only a series of steps, performed one after the other, can be an algorithm. A single–step procedure for performing a task is simply a restatement of the task, not an algorithm. Thus, a judge or

84. All pre–*Williamson* § 112(f) software cases have employed the strong presumptions in the threshold test. *WMS Gaming*—the case in which the Federal Circuit first grappled with the application of § 112(f) to software—was decided only five years before the Federal Circuit articulated the strong presumptions, and there were no cases in that five–year window that employed scope theory to animate the § 112(f) threshold test in software cases. *See* WMS Gaming Inc. v. Int'l Game Tech., 184 F.3d 1339 (Fed. Cir. 1999).

85. *See supra* notes 49–55 and accompanying text.

examiner cannot ask whether an individual claim limitation recites structure in the form of an algorithm.

Rather, if the plan is to carry the Federal Circuit's algorithm–as–structure analysis over from its precedent on identifying corresponding structure in a specification and use it to identify sufficient structure in a claim, judges and examiners must break with precedent in a different way and ask whether a claim stating a series of functional limitations constitutes an algorithm.[86] This approach to identifying structure in the threshold test would require a significant doctrinal shift, as the threshold test usually examines whether each functional limitation recites sufficient structure on its own, not whether a series of functional limitations collectively recites sufficient structure. In fact, even when a claim states that the same "means" performs more than one function, each function is supposed to be analyzed independently to determine whether it has sufficient structure to evade the strictures of § 112(f).[87] However, searching for structure in an aggregate of functional limitations is the only feasible approach if the Federal Circuit's well–established algorithm analysis is to be used to answer the definitional "What is structure?" question of the threshold test in software claims.

Yet, even if one is willing to alter the search for structure in this software–specific manner, the Federal Circuit's algorithm case law is decidedly unhelpful in the threshold test. The crux of the problem is that the Federal Circuit has only defined an algorithm in a relative manner. More specifically, it has only defined an algorithm in the specification in relation to a particular claim limitation. A specification's functional description of a software invention is an algorithm if the specification describes how the software operates in a manner that is *more granular* than the baseline functional description of the software in the claim limitation.[88] One cannot identify an algorithm under the Federal Circuit's definition without first

86. This question assumes that the claim recites more than one functional limitation. If a claim recites only a single limitation and that limitation is functional, then the claim is invalid. Section 112(f) only saves functional limitations in combination claims from invalidity under *Halliburton Oil Well Cementing Co. See In re* Hyatt, 708 F.2d 712, 712–15 (Fed. Cir. 1983). However, a claim may recite a single functional software limitation in combination with one or more non–functional limitations. It is not clear that the single functional limitation in such a claim could ever be held to recite sufficient structure to avoid § 112(f) when structure is equated with an algorithm.

87. *See, e.g.*, Media Rights Techs. v. Capital One Fin. Corp., 800 F.3d 1366, 1373 (Fed. Cir. 2015); Noah Sys., Inc. v. Intuit Inc., 675 F.3d 1302, 1318–19 (Fed. Cir. 2012).

88. *See supra* notes 57–59 and accompanying text.

having identified the baseline task in the claim that the algorithm is supposed to perform.

To appreciate the relative nature of the Federal Circuit's definition of an algorithm, consider the impossibility of examining a specification in isolation from the claims and meaningfully opining on whether it describes corresponding structure. The exact same specification can provide corresponding structure for one claim and yet fail to provide corresponding structure for another claim. For example, consider a hypothetical patent on a digital rights management technology.[89] Assume that the specification states that the invention performs three functions: it controls a client system's data output by diverting a commonly used data pathway of a media player application to a controlled data pathway, monitors the controlled data pathway, and disrupts media content playback at the controlled data pathway when playing the media file content is outside of the usage restriction applicable to the media file.[90] Is this functional description of how the invention works an algorithm? The answer depends on the claim at issue. Assume that claim 1 recites a single, umbrella limitation such as "a compliance mechanism for ensuring that only media content within the usage restriction applicable to the media is played." Here, the specification likely does describe structure in the form of an algorithm because it provides a series of more granular steps that explain how to perform the less granular task recited as the claim limitation. Now, assume that claim 2 recites three separate functional limitations that parallel the three disclosed functions. That is, assume that claim 2 recites separate "mechanism for controlling," "mechanism for monitoring," and "mechanism for disrupting" limitations. The unchanged specification no longer describes structure or an algorithm. Rather, it merely restates the functional tasks that are recited as claim limitations.[91] Under the Federal Circuit's relational definition of an algorithm, the existence of an algorithm, and thus corresponding structure, is contingent on the baseline of the functional description provided by the claims. A judge or examiner cannot look at a description of software functionality, standing alone, and reach the conclusion that the description describes structure.

89. In this hypothetical, the technology and language, but not the legal analysis, is based on the patent at issue in *Media Rights Technologies v. Capital One Financial Corp*, 800 F.3d 1366 (Fed. Cir. 2015).

90. *Cf. id.* at 1369–70.

91. *See supra* note 57 and accompanying text (noting that a restatement of the functional claim limitations is not an algorithm).

The Federal Circuit's relational definition of corresponding structure for software limitations does not map neatly onto the nature of corresponding structure in other technologies. In the mechanical arts, the answer to the definitional question is never relational or contingent. Any given property of a mechanical invention is either a structural or a functional property; a three–legged stool is not structure with respect to one claim but function with respect to another. There may be relativity when determining whether any given structure in the specification can perform the claimed function. It may well be that a three–legged stool is corresponding structure for a "means for supporting" limitation, but not for a "means for cutting" limitation. However, there is no relativity in the more fundamental definitional question of what constitutes structure in the first place.

The Federal Circuit's relational definition of structure in software is conceptually workable, if a bit awkward, when it is put to work in the context of identifying corresponding structure in the specification during claim construction. Because the search for structure in the specification is always conducted after a claim limitation has been held to be subject to § 112(f), the functional description recited in the claim is always available to serve as a baseline.[92] However, when searching for sufficient structure in

92. The Federal Circuit's relational definition of structure in software is awkward even when looking for corresponding structure because it places a formalistic, rather than substantive, limit on permissible patent scope. A relational definition of structure does not cap permissible claim scope at any particular level of generality, but instead mandates that every functional claim limitation be narrowed to a disclosed algorithm and its equivalents, regardless of the level of specificity at which the limitation is drafted. *See* Collins, *supra* note 8, at 1463–67. In an ideal patent regime that imposes substantive limits on what can be patented, two claims that are coextensive—that is, two claims that describe the same set of technologies—should be subjected to the same restrictions because they raise identical policy concerns. However, when operating on the basis of a relational definition of corresponding structure, § 112(f) does not achieve this goal. For example, if a first claim limitation recites function A and the specification recites algorithmic steps 1, 2, and 3 for function A, the claim containing the limitation is valid only if limitation A is restricted to the "structure" of steps 1, 2, and 3, as well as its equivalents. But, if a second claim were to directly recite functions 1, 2, and 3—which are identical to steps 1, 2, and 3—as separate limitations, § 112(f) would also require that each limitation in this claim be restricted in scope to the more granular, algorithmic steps (or sub–algorithmic steps, depending on your perspective) disclosed in the specification. Functions that are corresponding structure when recited as the steps of an algorithm in the specification cease to be structure when they are recited as a series of claim limitations. The limitation reciting function 1 would need to be restricted to something like the "structure" of sub–steps 1a, 1b, 1c, and its equivalents. To be clear, even the formalistic limitation on permissible patent scope that follows from the Federal Circuit's relational definition of an algorithm in § 112(f) does important work when software claims employ extremely general functional limitations. In some situations, however, it may not do enough work. It may narrow an extremely general claim to a very

a claim limitation as part of the § 112(f) threshold test, the relational definition of structure is conceptually bankrupt. There is no baseline to which to compare the claim limitation, and a relative definition is useless without a baseline. Just as it is impossible to know whether functional language in a specification constitutes an algorithm without consulting the baseline functional task provided by the claims, it is impossible to know whether functional language in a claim constitutes an algorithm without consulting a baseline task of some kind. The claim cannot provide this baseline. It is nonsensical to ask whether the claim limitations are steps in a step–by–step procedure for performing the functions specified in the claim limitations.[93]

When a trained patent lawyer looks at a software claim with a series of functional limitations, she may initially believe that she can use the concept of an algorithm to identify structure in that series of limitations during the § 112(f) threshold test. For example, presented with the hypothetical claim 1 (including the limitation "a compliance mechanism for ensuring that only media content within the usage restriction applicable to the media is played") and claim 2 (including limitations reciting "mechanism for controlling," "mechanism for monitoring," and "mechanism for disrupting"), she may see structure in claim 2 but not claim 1. The underlying intuition is that the limitations of claim 2 form an algorithm for performing the task listed in claim 1. But, this intuition adopts the function recited in claim 1 as the baseline task to be performed without any justification for doing so. Every series of functional limitations is a step–by–step process for doing something. Unless every series of functional limitations in a software claim recites an algorithm and thus sufficient structure to avoid the application of § 112(f), the pre–*Williamson* definition of an algorithm cannot be used to identify sufficient structure in the post–*Williamson* threshold test.

B. PRECEDENT ON THE PRE–*WILLIAMSON* THRESHOLD TEST

Before *Williamson*, the Federal Circuit occasionally entertained arguments addressing whether the amount of structure in a claim limitation was sufficient to rebut the strong presumptions of the § 112(f) threshold test. In principle, the reasoning in these arguments could provide insight

general, but still overbroad, claim. In other situations, it may counterproductively do too much work. It may further narrow an already sufficiently narrow claim.

93. *But cf. infra* note 121 and accompanying text (considering the possibility that a claim could recite both a task and a series of steps for accomplishing that task).

into a definition of software's structure for the post–*Williamson* threshold test. In actuality, however, they do not. Some pre–*Williamson* opinions use conclusory reasoning, baldly asserting a generic, programmed computer is sufficient structure to avoid § 112(f).[94] As explored below, other pre–*Williamson* opinions employ reasoning that is either misguided or protean. Subsection V.B.1 notes, and dismisses as unwise, the pre–*Williamson* cases in which software's structure is defined in terms of physical structure. Subsection V.B.2 turns to a sparse line of cases suggesting that functional claim limitations describing an operation's input, output, connections, or how its output may be achieved should count as logical structure. This definition of software's structure is radically underdeveloped, but it hints at what would be needed to formulate a stand–alone definition of software's logical structure for the post–*Williamson* § 112(f) threshold test.

1. Circuit as Physical Structure

A number of the Federal Circuit's pre–*Williamson* cases which found sufficient structure in a claim limitation lacking the term "means" to adhere to the presumption against the application of § 112(f) in the threshold test derive from a single precedent: *Apex Inc. v. Raritan Computer, Inc*.[95] The *Apex* claim contained a "programmed logic circuit" limitation for performing a variety of functions,[96] and the Federal Circuit concluded that the presumption against § 112(f) had not been overcome because a circuit requires a "conducting path" and thus entails physical structure: "The term 'circuit' is defined as 'the combination of a number of electrical devices and conductors that, when interconnected to form a conducting path, fulfill some desired function.'"[97] Citing *Apex* as precedent, the Federal Circuit has found enough structure in a number of other software limitations lacking the word "means" to conclude that the presumption against the application

94. *See, e.g.,* LG Elecs., Inc. v. Bizcom Elecs., Inc., 453 F.3d 1364, 1372 (Fed. Cir. 2006) ("The claim itself provides sufficient structure, namely 'a CPU and a partitioned memory system,' for performing the stated function, 'controlling the communication unit.'"), *rev'd on other grounds sub nom.* Quanta Comput., Inc. v. LG Elecs., Inc., 553 U.S. 617 (2008).

95. Apex Inc. v. Raritan Comput., Inc., 325 F.3d 1364, 1373–74 (Fed. Cir. 2003).

96. *Id.* at 1368.

97. *Id.* at 1373. For additional definitions of "circuit" that require an electrical pathway, see *Linear Technology Corp. v. Impala Linear Corp.*, 379 F.3d 1311, 1320 (Fed. Cir. 2014).

of the § 112(f) threshold test had not been overcome. Some of these limitations used the term "circuit,"[98] but others did not.[99]

Whatever merit this invocation of the physical structure of a circuit has when patent claims describe relatively simple hardware devices, it should be irrelevant to a definition of software's structure in a post–*Williamson* threshold test based on scope theory. The arbitrary nature of the Federal Circuit's "circuit" doctrine is on full display in *MIT v. Abacus Software*.[100] The patent at issue contained two functional limitations: an "aesthetic correction circuitry" for performing a function[101] and a "colorant selection mechanism" for performing a function.[102] Addressing the former limitation, the Federal Circuit identified enough structure for the presumption against § 112(f) not to be overcome. "[T]he term 'circuitry,' by itself, connotes sufficient [physical] structure," and, laundering function into additional structure, the "aesthetic correction" language "adds further structure by describing the operation of the circuit."[103] In contrast, the presumption against § 112(f) was overcome for the latter limitation. "The term 'mechanism' standing alone connotes no more structure than the term 'means,'" and, refusing to launder function into structure, the "colorant selection" modifier did not have a generally understood meaning in the art that connoted structure, either.[104] Here, the Federal Circuit's "circuit" doctrine is simply an example of "the doctrine of magic words" in action.[105] There is nothing that is meaningfully more structural about the "aesthetic

98. *See, e.g.*, *id.* at 1320–21; Power Integrations, Inc. v. Fairchild Semiconductor Int'l, Inc., 711 F.3d 1348, 1365 (Fed. Cir. 2013); MIT v. Abacus Software, 462 F.3d 1344, 1353–55 (Fed. Cir. 2006).

99. Inventio AG v. ThyssenKrupp Elevator Am. Corp., 649 F.3d 1350, 1358 (Fed. Cir. 2011).

100. *MIT*, 462 F.3d at 1353–57.

101. *Id.* at 1348 (describing an "aesthetic correction circuitry for interactively introducing aesthetically desired alterations into [the] appearance signals to produce modified appearance signals").

102. *Id.* (describing a "colorant selection mechanism for receiving said modified appearance signals and for selecting corresponding reproduction signals . . . to produce . . . a colorimetrically-matched reproduction").

103. *Id.* at 1355–56. The dissent argued that the limitation was a § 112(f) limitation. *Id.* at 1360–65 (Michel, C.J., dissenting). The majority's rejoinder fell back on the strong presumptions. *Id.* at 1356 ("[T]he dissent appears to misapprehend the strength of the presumption that applies when the term 'means' does not appear in the claim.").

104. *Id.* at 1353–55.

105. Julie E. Cohen & Mark A. Lemley, *Patent Scope and Innovation in the Software Industry*, 89 CALIF. L. REV. 1, 9 (2001) (describing a rule under which software was patentable so long as the applicant recited "magic words and pretended that she was patenting something else entirely").

correction circuitry" limitation than the "colorant selection mechanism" limitation.

2. *Inputs and Outputs as a Clue to Logical Structure*

In his majority opinion in *Apple v. Motorola*, issued only a year before *Williamson*, Judge Reyna held that a functional software limitation without the word "means" recited enough structure so that the presumption against § 112(f) was not overcome because it described the "input" and "output" of the software, as well as "how its output may be achieved."[106] With this reasoning, Judge Reyna laid some preliminary groundwork for a stand–alone definition of logical structure in software—a definition that can be used to identify structure in a functional description of how software works without employing some other functional description as a baseline.

Among the many patents and issues addressed, *Apple v. Motorola* upheld several apparatus claims on the use of finger contacts to change the visual field or select files on a touchscreen computer.[107] Each of the claims described one or more "heuristics" for determining that certain finger contacts correspond with certain commands.[108] The district court construed the term "heuristics" to mean "one or more rules to be applied to data to assist in drawing inferences from that data."[109] It then held the heuristics limitations to be subject to § 112(f) under the threshold test, despite the then–present strong presumption to the contrary, because the claims described functions "without describing the structure necessary to perform the functions."[110]

On appeal, Judge Reyna's majority opinion reversed and held that the heuristics limitation was not subject to § 112(f) because the limitation recited "the heuristics' operation within the context of the invention, including the inputs, outputs, and how certain outputs are achieved."[111] In

106. Apple Inc. v. Motorola Inc., 757 F.3d 1286, 1300 (Fed. Cir. 2014).

107. *Id.* at 1294–1304. The *Apple* opinion is much better known for its implications for injunctions for standard–essential patents. *See id*. at 1331–32.

108. *Id.* at 1295 (noting, for example, that one limitation recited "a vertical screen scrolling heuristic for determining that one or more finger contacts correspond to a one-dimensional vertical screen scrolling command rather than a two-dimensional screen translation command *based on an angle of initial movement of a finger contact with respect to the touch screen display*").

109. *Id.* Judge Posner sat by designation as the district court judge. *Id*. at 1294.

110. *Id.* at 1295.

111. *Id.* at 1301; *see also id.* at 1299 ("Structure may . . . be provided by describing the claim limitation's operation, such as its input, output, or connections."); *id.* at 1300 (stating that the heuristics limitation "describes the limitation's operation, including its

some ways, this inputs–and–outputs line of reasoning is problematic. Its support in Federal Circuit precedent is weak.[112] More importantly, the claim at issue arguably did not actually include any reference to how outputs are achieved as a limitation on claim scope. Judge Reyna relied on features of the preferred embodiments disclosed in the specification that do not actually limit claim scope to find that the claim language connoted sufficient structure.[113] However, abstracting from the actual facts of the case, Judge Reyna's opinion is interesting because it suggests one factor that might be relevant in developing a stand–alone definition of software's structure. The gist of Judge Reyna's reasoning is that functional claim language should count as a description of the logical structure of software if it is sufficiently specific that it describes how the invention works—that is, if it describes "inputs, outputs, and how certain outputs are achieved"—rather than what the invention does for its user.[114] This approach to answering the threshold

---

input, output, and how its input may be achieved"). Judge Reyna also argued "that 'heuristic' has a known meaning" that connotes structure, but he eventually concluded that "[w]e need not decide here whether the term 'heuristic,' by itself, connotes sufficient structure to maintain the presumption against means-plus-function claiming" because the inputs–and–outputs argument was sufficient to demonstrate that the strong presumption against § 112(f) had not been rebutted. *Id.* at 1300–01.

112. *Apple* cites two cases to support looking to the "input, output, and connections" of a program as a clue for identifying structure in a claim limitation. *Id.* at 1299. The first case, *Linear Technology Corp. v. Impala Linear Technology Corp.*, held that the presumption against § 112(f) was not overcome principally because the term "circuit" connoted tangible structure. 379 F.3d 1311, 1320–21 (Fed. Cir. 2014); *see also supra* Subsection V.B.1. However, it did also note in passing that the term "circuit" became more structural when coupled in the claim with a description of the circuit's "operation" and its "desired output." *Linear*, 379 F.3d at 1320. The second case, *Lighting World, Inc. v. Birchwood Lighting, Inc.*, held that the presumption against § 112(f) was not overcome principally because the term "connector" had a generally understood meaning in the art as a name for physical structure. 382 F.3d 1354, 1361–62 (Fed. Cir. 2004).

113. The claim language clearly recited the heuristic's "objectives." *Apple*, 757 F.3d at 1302. Beyond that, however, all of the opinion's support for the inputs, outputs, and connections as claim limitations drew from the written description. *Id.* at 1302–03. One of the dissent's critiques in *Apple* was that the majority relied too heavily on the preferred embodiments in the specification to identify structure in the claims. *Id.* at 1335 (Prost, J., concurring in part and dissenting in part). This type of disagreement should not be surprising given that the Federal Circuit has not even been clear about whether *Phillips* governs the search for sufficient structure in § 112(f) threshold test. *See supra* note 81.

114. *Apple*, 757 F.3d at 1301. The language used to convey this reasoning, however, is at times misleading. As the opinion puts it, "[t]he limitation's operation is more than just its function; it is how the function is achieved in the context of the invention." *Id*. at 1299. This passage overlooks the fact that a description of "how the function is achieved in the context of the invention" is itself inevitably a description of function. *See supra* note 65 and accompanying text (noting that the Federal Circuit sometimes misleadingly juxtaposes

test's definitional question is underdeveloped, but it at least hints at a way of defining software's structure that could work in the post–*Williamson* § 112(f) threshold test.[115]

VI. WHICH REVOLUTION?

Williamson's shift to a threshold test based on scope theory mandates a revolution in the law of functional claiming as applied to software because pre–*Williamson* opinions are of little use when identifying software's structure as part of the § 112(f) threshold test. However, it is not yet clear which of a number of possible revolutions *Williamson* will yield. The remainder of this Essay considers four possibilities. The first three stick with the Federal Circuit's relational definition of an algorithm and attempt to resolve the problem that this definition creates in different, unsatisfying ways.[116] The fourth possible revolution charts a new course: following the lead of *Apple v. Motorola*, the Federal Circuit could articulate a new, stand–alone definition of software's structure for the purposes of § 112(f).

A. THREE UNDERWHELMING REVOLUTIONS

A first possible revolution is that all software claims with multiple functional limitations will be held to recite algorithms and avoid the strictures of § 112(f). All claims with multiple functional limitations recite an algorithm insofar as they present a series of steps for accomplishing the task of providing the patented technology's end–user with some utility.[117] However, this outcome is unlikely in light of the expectation that a tilt

algorithms with functional descriptions). Judicial opinions would be more conceptually accurate if they were to say that a description of inputs, outputs, and how certain outputs are achieved is a type of functional description that counts as metaphorical structure in software because it serves the needed policy of limiting claim scope.

115. At least one district court has used *Apple*'s inputs–and–outputs reasoning to find sufficient structure in a claim limitation lacking the word "means" to avoid § 112(f) status after *Williamson*. *See* Finjan, Inc. v. Proofpoint, Inc., No. 13-cv-05808-HSG, 2015 WL 7770208, at *11 (N.D. Cal. Dec. 3, 2015) ("The term 'content processor' has a sufficiently specific structure. Independent claim 1 describes how the 'content processor' interacts with the invention's other components (the transmitter and receiver), which informs the term's structural character . . . . [T]he intrinsic evidence establishes the structural character of 'content processor' through its interaction with the system's other components.").

116. The first and third of these possible revolutions must accept that the search for software's structure does not require a separate search for sufficient structure within each functional limitation but rather a search for structure in a series of limitations. *See supra* notes 85–87 and accompanying text.

117. Claims with only a single functional limitation would not recite structure under this approach. *Cf. supra* note 86 (noting the invalidity of single–means claims).

toward scope theory in the threshold test should increase, not decrease, the number of software limitations subject to § 112(f).

A second possible revolution inverts the outcome of the first: perhaps all functional software limitations will be governed by § 112(f). A claim cannot be an algorithm for itself[118] So, perhaps all functional software limitations will be labeled as structureless tasks and no functional software limitations will connote sufficient structure. This route forward satisfies the expectation that the turn to scope theory in *Williamson* should increase the number of software limitations subject to § 112(f). In fact, it would lead to a tidal shift in the law of software patents. Before *Williamson*, most functional software limitations using "nonce" words were not subject to § 112(f), but after *Williamson*, all such limitations would be.[119]

A third possible revolution amounts to a mandate for a new style for drafting software claims. Perhaps a software claim recites an algorithm, and thus sufficient structure, if it initially recites both a general task as a limitation and then a more specific, step–by–step process for achieving that task. This approach bootstraps the baseline task that is needed to use the concept of an algorithm to identify structure into an earlier part of the claim. To continue the hypothetical addressed above, consider a claim to a digital rights management technology reciting "mechanism for controlling," "mechanism for monitoring," and "mechanism for disrupting" as separate limitations.[120] Imagine that the claim also includes a statement of the overall task to be performed, perhaps as part of the preamble, such as "a compliance mechanism for ensuring that only media content within the usage restriction applicable to the media is played." Now, a relative definition of an algorithm could get some traction in the threshold test by comparing two different parts of the same claim. Perhaps claims that explicitly state a baseline task contain structure in other limitations if those other limitations constitute a step–by–step procedure for performing that task.[121]

118. *See supra* text accompanying note 93.

119. This revolution would extend the formalism of the pre–*Williamson* law: all claims would be pushed down one step on the ladder of generality, regardless of whether they are initially drafted in a general or specific manner. *See supra* note 92.

120. *See supra* notes 90–91 and accompanying text.

121. In substance, this third revolution in the required form of a functional software claim is not much different from the second revolution under which every functional software limitation is subject to § 112(f). Rather than narrowing the scope of the broad, functional limitation to the algorithm disclosed in the specification (and its equivalents), it narrows the scope of the claim to the algorithmic steps recited as claim limitations.

B. A FOURTH REVOLUTION: DEVELOP A STAND–ALONE DEFINITION OF SOFTWARE'S STRUCTURE

A fourth possible revolution takes a decidedly different tack on the problem. Following the lead of *Apple v. Motorola*, the Federal Circuit could develop the stand–alone definition of software's structure that is needed for the threshold test to draw a principled line that subjects some, but not all, functional software limitations to § 112(f) based on the limitations' absolute breadth. In *Williamson*, the Federal Circuit recognized that the policy concerns about overbreadth expressed in the Supreme Court's functional–claiming cases from the first half of the twentieth century require a threshold test motivated by scope theory.[122] When there is enough structure recited in a claim's limitations, the number of "other devices beyond our present information or indeed our imagination which will perform [the claimed] function and yet fit [the] claims" is sufficiently reduced to ward off the most serious social costs of functional patents.[123] The same policy concerns should frame a stand–alone answer to the sui generis definitional "What is structure?" question of the threshold test in software. At what level of claim specificity are the restraints on competition engendered by functional software claims acceptable because the claim is restricted in scope to particular means for achieving a useful end?

The basic premise of a stand–alone definition of structure in software must be a line on a ladder of generality or abstraction. Functional descriptions of a software invention exist at many different levels of generality, ranging from extremely specific (e.g., an in–depth description of modules and their interaction) to extremely general (e.g., a description of the utility enjoyed by the end–user of the software). A stand–alone definition of structure must identify the point on the ascent of this ladder at which a granular description of a software program that constitutes logical structure transitions into a highly general, structureless description of a software program.[124] The descriptions both above and below the line are technically functional descriptions in a linguistic sense, but descriptions at

122. *See supra* notes 16–17 and accompanying text.

123. Halliburton Oil Well Cementing Co. v. Walker, 329 U.S. 1, 12 (1946).

124. In a sense, the term "stand–alone structure" is misleading. The identification of structure still requires a comparison to a baseline: Is the functional description above or below the point at which the transition occurs? However, this baseline is fixed and always available. Unlike a comparison to the baseline provided by a claim, this baseline does not shift dramatically as the claim language varies or disappears altogether when the search focuses on finding structure in that very claim.

or below the line count as metaphorical or logical structure for the purposes of § 112(f).

In his work on functional claiming in software, Mark Lemley alludes to a level–of–generality analysis when he argues that § 112(f) must make a distinction between a "goal" (not structure) and a "way of implementing a goal" (structure),[125] or a "problem" the patentee solved (not structure) and "what the patentee . . . actually did" to solve the problem (structure).[126] The difficulty with these distinctions is that goals and problems can be expressed in functional language at many different rungs on the ladder of abstraction. Your "goal" is likely to be part of my way of implementing a different "goal." A software inventor may be motivated by the "problem" that two modules interact in an inefficient manner or by the "problem" that the parties to a transaction face settlement risk.[127] What is needed in the post–*Williamson* world is a stand–alone definition of what constitutes a general, structureless problem or goal and, inversely, what constitutes a structural way of implementing a goal or a patentee's actual solution to a problem. Simply defining an implementation in relation to a goal is not enough if what constitutes a goal remains a moving target.

Ideally, a stand–alone definition of structure should govern all inquiries into the structure of a software invention under § 112(f). *Williamson* specifically addresses only the § 112(f) threshold test, so the most immediate repercussion of *Williamson* is understandably a need for a new definition of software's structure that works in this context. However, once a stand–alone definition of structure has been developed, there is no reason not to use it to identify the corresponding structure of § 112(f) limitations in the specification, as well.[128]

---

125. Lemley, *supra* note 8, at 947.

126. *Id.* at 963.

127. *Cf.* Alice Corp. v. CLS Bank Int'l, 134 S. Ct. 2347 (2014) (holding that a claim to a computer–executed method of reducing settlement risk is patent–ineligible). The interplay between the Federal Circuit's *Williamson* opinion and the Supreme Court's *Alice* opinion raises interesting questions that are beyond the scope of this Essay. Nonetheless, it is worth noting that a stand–alone definition of software's structure developed to implement *Williamson* might shed some light on when a claim "purport[s] to improve the functioning of the computer itself," and thus when a claim is patent–eligible even if it is directed to an abstract idea. *Id.* at 2359. Perhaps improvements in logical structure are examples of improvements in the functioning of the computer itself.

128. Technically, the Federal Circuit could adopt a stand–alone definition of structure for the threshold test and continue to use its relational definition of structure to identify corresponding structure in the specification. However, once the stand–alone definition has been formulated, it makes little sense not to use it in both contexts. A stand–alone definition

Drawing the line that is needed to identify logical structure in the software arts will not be an easy task. The echoes of Learned Hand's levels–of–generality test for drawing the idea/expression dichotomy in copyright are clear, and that test is notorious for its lack of ex ante clarity.[129] In a yet closer parallel, the Federal Circuit shelved the project of bringing § 112(f) to bear on step–plus–function limitations in method claims, despite the statutory requirement to do so, because of the difficulty of developing a stand–alone definition of an "act."[130] In method claims, there is no clear, intuitive difference between those actions that constitute steps and those that constitute acts.[131] The only difference between them is the position of a functional description of an action on a ladder of generality. An act describes a more specific functional task than a step does. When is a functional operation recited as a claim limitation sufficiently specific to qualify as an act rather than a step? The Federal Circuit punted on this question because of its difficulty.[132] Yet, this question closely parallels the question that the Federal Circuit must answer in order to create a stand–alone definition of software's structure.

The line marking a binary legal distinction between general and specific software–implemented functions on the ladder of abstractions has no "natural" position determined by extra–legal concerns. Nor is it feasible to ask the ultimate economic question—at what level of abstraction does a

---

of structure in the context of the search for corresponding structure in the specification would eliminate the formalism of lowering the permissible level of generality of a functional claim by one rung regardless of the claim's initial breadth. *See supra* note 92.

129. Nichols v. Universal Pictures Corp., 45 F.2d 119, 121 (2d Cir. 1930). The actual copyright doctrine that has developed to administer the idea/expression dichotomy in software cases is not helpful in patent law. In copyright, the idea/expression dichotomy seeks, among other goals, to prevent copyright from granting an author control over software functionality at any level of abstraction. Comput. Assocs. Int'l v. Altai, Inc., 982 F.2d 693 (2d Cir. 1992). In patent law, § 112(f) must draw a line between claims to functionality that are sufficiently specific and those that are excessively general.

130. *See supra* notes 28–30 and accompanying text.

131. In an extended concurrence addressing step–plus–function claims, Judge Rader recognized that the distinction between a step and an act is "inherently more problematic" than the distinction between a function and a structure, at least in non–software claims. Seal-Flex, Inc. v. Athletic Track & Court Const., 172 F.3d 836, 848–49 (Fed. Cir. 1999) (Rader, J., concurring).

132. Judge Rader justified this punt by leaning heavily on intent theory to create an even stronger–than–usual strong presumption in the § 112(f) threshold test that method claim limitations without the (rarely used) term "step" are not step–plus–function limitations. *Id.* at 849. Yet, if *Williamson* applies to both apparatus and method claims, the Federal Circuit can no longer rely on a strong presumption against the application of § 112(f) to avoid drawing a distinction between steps and acts in method claims.

particular patent claim provide sufficient, but not excessive, reward to a particular software innovator?—as part of each and every patent examination at the Patent and Trademark Office and invalidity defense in the courts. What is needed is an administrable proxy for the ultimate economic question to replace the proxy of physical structure that is used in mechanical technologies but that is unavailable in software. I have suggested elsewhere that one possible proxy uses consumer preferences as an anchor.[133] Perhaps descriptions of functionality that map onto the reasonable consumer's desires should be labeled as goals because these preferences contribute to the definition of product markets, and perhaps descriptions of functionality that map onto technological ways of satisfying those desires should be seen as ways of achieving those goals and thus as logical structure. This proxy is far from straightforward, but it at least provides a stable point of reference to which the courts can peg a stand–alone definition of software's structure.

The next step to develop a stand–alone definition of structure for software under § 112(f) should involve an interdisciplinary conversation among patent lawyers, computer scientists, and economists.[134] The desired output of such a conversation would be a menu of stable levels of generality, grounded in computer science, at which all software can be described, and an analysis of the implications of selecting any one of these levels as a stand–alone definition of logical structure in software for the purpose of § 112(f). Having identified a menu of options and clarified the consequences of each option, the Federal Circuit would be well positioned to lead the revolution in software patent law called for by *Williamson*. It is possible that the conversation will not ultimately produce the distinctions necessary to provide an acceptable answer to the definitional "What is structure?" question. At the end of the day, the lesson learned might be that there are, in fact, no viable distinctions upon which to build a reformed law of software patents with the desired clarity. The purely functional nature of software may mean that § 112(f) just cannot regulate software patents in the

133. Collins, *supra* note 8, at 1421–23, 1466.

134. *See id.* at 1466–67 (arguing that such a conversation is needed). Professor Pamela Samuelson's efforts at revamping intellectual property protection for computer software provide an interesting model of the needed conversation, but not of the desired outcome, because the scope of the task at hand after *Williamson* is quite different from the scope of the task that Professor Samuelson undertook. *See* Pamela Samuelson et al., *A Manifesto Concerning the Legal Protection of Computer Programs*, 94 COLUM. L. REV. 2308 (1994) (proposing a sui generis regime for intellectual property protection for software).

way that it regulates mechanical patents.[135] Nonetheless, the conversation would still be valuable. By taking a stand–alone definition of software structure off the table, it would bolster the case for a different revolution.

VII. CONCLUSION

Williamson v. Citrix Online altered the threshold test for determining whether a functional claim limitation that does not use the term "means" should be construed using the scope–narrowing rules of § 112(f). On its face, *Williamson* may only appear to implicate the quantitative "How much structure is enough?" question of the threshold test. That is, *Williamson* may only appear to require a bit more structure in the claim limitations than was previously required in order to evade § 112(f). However, with respect to software patents in particular, this Essay demonstrates that *Williamson* calls for a revolution in the law of functional claiming. *Williamson* will finally force the Federal Circuit to address the definitional "What is structure?" question as part of the § 112(f) threshold test in a more open and thorough manner.

The idea that *Williamson* will trigger a revolution in the legal definition of software's structure may be counterintuitive because, before *Williamson*, there was already a well–established answer to the definitional question in the context of identifying software's corresponding structure in the specification. Software's corresponding structure was an algorithm, or step–by–step procedure, for performing the function recited as a claim limitation. A revolution is needed, however, because this definition of an algorithm is relative and thus not well suited for identifying software's structure in the context of the threshold test that *Williamson* addresses. A search for an algorithm can identify structure in a functional description of a software program only in relation to the functional language employed in a claim limitation. The concept of an algorithm is not helpful when assessing whether the functional claim language itself recites structure: examining the functional language in a claim in relation to itself would be like staging a legal theater of the absurd. For this reason, *Williamson* requires a complete reconceptualization of what constitutes software's structure under § 112(f).

135. *Cf.* Kevin Emerson Collins, *Patent-Ineligibility as Counteraction*, 94 WASH U. L. REV. (forthcoming 2017) (arguing that purely functional technologies cause regulatory inefficacy in doctrines that, like § 112(f) and written description, rely on the recitation of structure to limit the overbreadth of functional claims).

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.

SOFTWARE PATENTS AS A CURRENCY, NOT TAX, ON INNOVATION

Colleen V. Chien[†]

ABSTRACT

Software innovation is transforming the U.S. economy. Yet our understanding of how patents and patent transactions support this innovation is limited by a lack of public information about patent licenses and sales. Claims about the patent marketplace, for example, extolling the virtues of intermediaries like non–practicing entities, or characterizing software patent licenses as a tax on innovation tend not to be grounded in empirical evidence. This Article brings much–needed data to the debate by analyzing transactional patent data from multiple sources and reporting several novel findings. First, this study finds that, despite reductions in the enforceability of software parents and levels of patent litigation, the market for software patents has remained remarkably robust, and actually grown in the number of transacted assets. The strength of this demand appears to be driven by the defensive—not only offensive—value of software patents, the importance of software–driven business models, and bargain shopping in the acquisition of patents. Second, this Article explores the extent to which software patent transfers support the transfer of technology as opposed to supporting just the transfer of liability, or freedom from suit, with mixed results. This study finds that the majority of material software licenses reported by public companies to the SEC from 2000–2015 (N=245) support true technology transfer. However, in recent years, large numbers of software patents apparently have also been sold to avoid litigation or to provide general operating freedom, rather than to access specific technologies. Software patents transferred between public companies from 2012 and 2015 were two to three times more likely to go from an older company to a younger company, and from a higher revenue to a lower revenue public company. These findings underscore the enduring importance of software patents in

DOI: https://dx.doi.org/10.15779/Z38TM7213Z

† Associate Professor, Santa Clara University School of Law, Former Obama White House Senior Advisor, Intellectual Property and Innovation. I thank David Schwartz, Michael Risch, Brian Love, Jorge Contreras, John Duffy, Jonathan Barnett, Damon Matteo, Pam Samuelson, Molly Van Houweling, Dan Lang, Aaron Perzanowski and audiences at the 2016 Berkeley Center for Law and Technology–Berkeley Technology Law Journal Conference on Software Innovation as well as the Hoover IP2 Spring 2016 Conference for their comments, Theresa Yuan, Noah Weeks-Bank, Mike Kenstowicz, Campbell Yore, Max Looper, John McAdams, and Angela Habibi for their excellent research assistance, Christian Chessman and the editors at BTLJ for their attention to detail and patience on the Article, and Esmaeil Khaksari of Innography, as well as Cash McNeel and John Wiora of ktMINE for assistance with patent database searching and access. This research was supported by unrestricted grants from the Santa Clara University Law School Summer Research fund, the Hoover Institution at Stanford, and Google.

supporting both technology transfer and freedom to operate. Despite the prevalence of NPEs, most patents are not bought for assertion, but to support these critical innovation functions. As such, the data support the characterization of software patents as a currency of—rather than a tax on—innovation.

TABLE OF CONTENTS

Software is eating the world.[1]

– Marc Andreesen

I. INTRODUCTION

The same week that Marc Andreessen published his well–known 2011 essay, "Why Software is Eating the World," Google moved to buy handset–maker Motorola Mobility for $12.5 billion.[2] Andreesen cited this development and others—the rise of software companies like Amazon, Netflix, and Shutterfly and the demise of bricks–and–mortar companies like Borders, Blockbuster, and Kodak—for the proposition that software had disrupted or would be disrupting industries across the economy, and would require companies to adapt to new, digitally–driven business models, or die. Since then, his prophecy has played out in numerous sectors of the economy— including automobiles, aerospace and defense, medical devices, and pharmaceuticals—as firms increasingly turn to software to differentiate products, enhance product performance, and increase user utility.[3] But just as Google's acquisition underscored the dominance of new, digital companies, it also demonstrated the importance of an instrument that has existed for over two hundred years,[4] the U.S. patent. Because while Google acquired Motorola's physical assets through the deal, its main objective was to acquire Motorola's intangible assets—its patents.[5] As Google CEO Larry Page wrote in a blog post, Motorola's patents were key to protecting Google's Android operating system from potential attacks by competitors like Microsoft, Apple, and others.[6]

---

1. Marc Andreessen, *Why Software is Eating the World*, WALL ST. J. (Aug. 20, 2011), http://online.wsj.com/article/SB10001424053111903480904576512250915629460.html.

2. *Id.*; Evelyn M. Rusli & Clair Cain Miller, *Google to Buy Motorola Mobility for $12.5 Billion*, N.Y. TIMES: DEALBOOK (Aug. 15, 2011, 7:43 AM), http://dealbook.nytimes.com/2011/08/15/google-to-buy-motorola-mobility/.

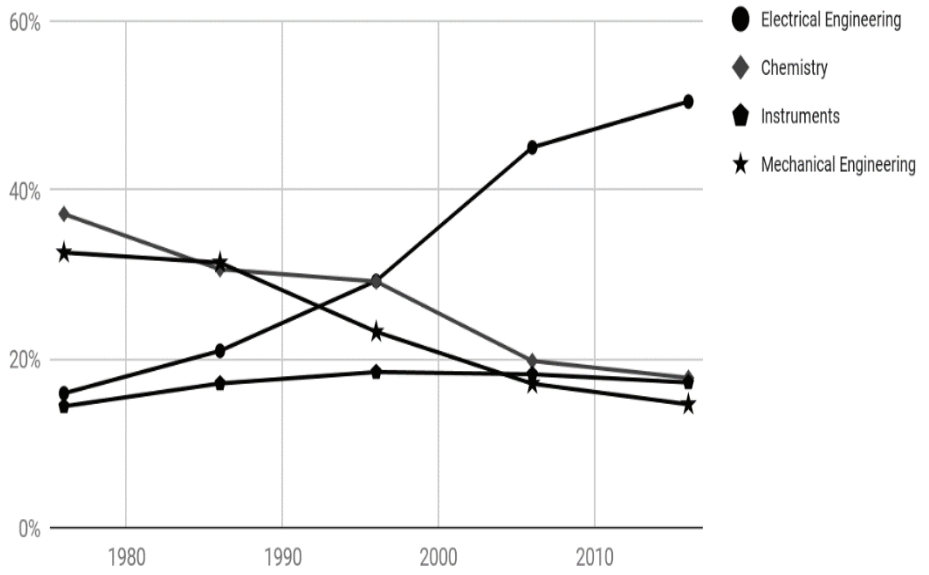
3. *See* Lee Branstetter, Matej Drev & Namho Kwon, *Get with the Program: Software-Driven Innovation in Traditional Manufacturing* (Nat'l Bureau of Econ. Research, Working Paper No. w21752, 2015), https://ssrn.com/abstract=2699996.

4. The first era of U.S. patenting was from 1790 to 1793, and resulted in few issuances. *See* EDWARD C. WALTERSCHEID, TO PROMOTE THE PROGRESS OF USEFUL ARTS: AMERICAN PATENT LAW AND ADMINISTRATION, 1798–1836 at 259–64 (1998).

5. *See, e.g.*, WALTER ISAACSON, THE INNOVATORS: HOW A GROUP OF HACKERS, GENIUSES, AND GEEKS CREATED THE DIGITAL REVOLUTION (2014).

6. Larry Page, *Supercharging Android: Google to Acquire Motorola Mobility*, OFFICIAL GOOGLE BLOG (Aug. 15, 2011), https://googleblog.blogspot.com/2011/08/supercharging-android-google-to-acquire.html.

Figure 1: Shares of U.S. Patents by Industry (1970–2016)[7]



Just as software innovation is on the rise, so is U.S. software patenting. Identifying software patents is notoriously difficult, but applying the World Intellectual Property Organization's industry definitions, the share of U.S. patents that can be classified under "Electrical Engineering"—a class that includes digital communications, computer technology, and communications, among others[8]—has grown markedly. In 1975, about 15%

7. Colleen V. Chien, *Opening the Patent System: Diffusionary Levers in Patent Law*, 89 S. CAL. L. REV. 4 (2016) (relying on data from the European Patent Office Worldwide Patent Statistical Database (PATSTAT) and taxonomy from the World Intellectual Property Organization (WIPO)); *accord* Alan C. Marco et al., *The USPTO Historic Patent Data Files: Two Centuries of Invention* (U.S. Patent and Trademark Office, Econ. Working Paper No. 2015-1, 2015), https://www.uspto.gov/sites/default/files/documents/USPTO_economic_WP_2015-01_v2.pdf, at 37 fig.11 (showing that annual patent grants in the "Computers & Communications and Electrical & Electronics NBER categories vastly outnumber patents in all other categories beginning in the early 2000s").

8. For a description of the scheme, including a complete list of subclasses within "Electrical Engineering," and the rationale for the breakdown, see Ulrich Schmoch, *Concept of a Technology Classification for Country Comparisons: Final Report to the World Intellectual Property Office (WIPO)*, WORLD INTELL. PROP. ORG. (June 2008), http://www.wipo.int/export/sites/www/ipstats/en/statistics/patents/pdf/wipo_ipc_technology.pdf. This approach was developed later than the industry categorization developed by Bronwyn H. Hall, Adam B. Jaffe, and Manuel Trajtenberg, and is preferred for this reason. *See* Bronwyn H. Hall et al., *The NBER Patent Citation Data File: Lessons, Insights and Methodological Tools* (Nat'l Bureau of Econ. Research Working Paper No. 8498, 2001),

of all new U.S. patents were for electrical engineering applications, with no one industry grouping capturing a majority of patents. In 2015, the electrical engineering share rose to nearly 50% (Figure 1). The remaining industry segments—including instruments, chemicals (a category that includes pharmaceutical drugs), and mechanical engineering—divided most of the remainder roughly evenly (Figure 1).

The question is whether software is eating the world because of software patents, despite them, or for some other reason. Patents encourage investment and risk–taking in innovation by granting exclusive rights in exchange for novel, nonobvious inventions. But they can also interfere with downstream innovation by preventing others, including those who invent independently, from practicing their own inventions. Young companies experience these tradeoffs most acutely: when a startup gets a patent, its likelihood of funding rises,[9] as most small firms do not patent.[10] But if the company becomes the target of litigation, the event is highly disruptive and can cause the firm to pivot away from products lines[11] or reduce research and development ("R&D") expenditures.[12]

Whether these patent dynamics are at the periphery of software innovation or at the heart of it remains unclear. According to one view, the value proposition associated with software–based innovation is so compelling that such innovation will happen regardless of the initial distribution of rights, which can be altered by contract.[13] In the digital

---

http://www.nber.org/papers/w8498.pdf (identifying the earlier categorization method developed by Hall and others).

9. *See* Joan Farre-Mensa et al., *What Is a Patent Worth? Evidence from the U.S. Patent 'Lottery'* (USPTO Econs. Working Paper No. 2015-5, 2015), http://ssrn.com/abstract=2704028.

10. *See* Stuart J.H. Graham et al., *Business Dynamics of Innovating Firms: Linking U.S. Patents with Administrative Data on Workers and Firms* (USPTO Econs. Working Paper No. 2015-3, 2015), http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2637602.

11. *See, e.g.*, Colleen V. Chien, Startups and Patent Trolls (Sept. 13, 2012) (unpublished manuscript), http://digitalcommons.law.scu.edu/cgi/viewcontent.cgi?article=1554&context=facpubs; *see also* Colleen V. Chien, Startups and Patent Assertion (Sept. 2013) (unpublished manuscript), http://digitalcommons.law.scu.edu/facpubs/856/.

12. *See, e.g.*, Catherine E. Tucker, *Patent Trolls and Technology Diffusion* (TILEC Discussion Paper No. 2012-030, 2013), http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2136955; Fiona M. Scott Morton & Carl Shapiro, *Strategic Patent Acquisitions*, 79 ANTITRUST L.J. 463, 463 (2014) (finding the enhanced monetization of patents by patent assertion entities (PAEs) to be harmful to innovation); Roger Smeets, Does Patent Litigation Reduce Corporate R&D? An Analysis of US Public Firms (Apr. 2014) (unpublished manuscript), https://www.tilburguniversity.edu/upload/8f3507ab-df1f-46c5-89a4-e1855f171404_Main_Litigation.pdf.

13. Robert P. Merges, *Contracting into Liability Rules: Intellectual Property Rights and Collective Rights Organizations*, 84 CALIF. L. REV. 1293 (1996).

world, monopolies are driven not by the right to exclude conferred by patents, but by network effects, scale,[14] and winner–take–all economics.[15] But patents are hard to ignore when Google spends more money on them than on R&D, as it did in the year of the Motorola purchase.[16] So did Apple that year, when it contributed to the purchase of patents from defunct telecommunications equipment provider Nortel for $4.5 billion.[17] These sales were huge and anomalous, but also raise concerns about the vulnerability of those with fewer resources to buy protection or patents, which includes just about every other company.[18]

The controversy over software patents also extends to software patent transactions. Patent transactions can enhance the patent system's incentive–inducing role by supporting specialization and extending the reach of the patent system to those who invent regardless of their position in the marketplace, helping to overcome the advantages of incumbents.[19] A startup company's ability to license or sell, rather than develop their technology, reduces its market risks and enhances innovation through its transfer of technology.[20] Patents can support the diffusion of software innovation between firms by providing transferable, tradeable assets.

But the growth in software patent litigation, including by non–practicing entities ("NPEs") (also known as patent assertion entities or "trolls"), has also been supported by the patent marketplace.[21] In a 2011 report to

14. *See* PETER THIEL, ZERO TO ONE 3-5 (2014) ("What does a company with large cash flows far into the future look like? Every monopoly is unique, but they usually share some combination of the following characteristics: proprietary technology, network effects, economies of scale, and branding").

15. *See, e.g.*, Om Malik, *In Silicon Valley Now, It's Almost Always Winner Takes All*, NEW YORKER (Dec. 30, 2015), www.newyorker.com/tech/elements/in-silicon-valley-now-its-almost-always-winner-takes-all.

16. Based on public filings and data, in 2012, Google spent $12.5 billion to buy Motorola Mobility and its patents, and $5.2 billion on R&D. In 2011, Apple spent $2.4 billion on R&D but contributed more to purchasing patents, including an estimated $2.6 billion on a single transaction to buy patents from Nortel. *See* Colleen V. Chien, *Reforming Software Patents*, 50 HOUS. L. REV. 325, 329 nn.11 & 12 (2012).

17. *Id.*

18. Rochelle Cooper Dreyfuss & Lawrence S. Pope, *Dethroning* Lear*? Incentives to Innovate After* MedImmune, 24 BERKELEY TECH. L.J. 971, 975 (2009) (describing issues presented by defensive patenting for startups and "cash–starved" companies).

19. *See infra* Section IV.B.

20. Stuart J.H. Graham & Ted Sichelman, *Why Do Start-Ups Patent?*, 23 BERKELEY TECH. L.J. 1063, 1074–76 (2008) (describing innovative benefits and strategic reasons for startup companies to license patented products).

21. *See* EXEC. OFFICE OF THE PRESIDENT, PATENT ASSERTION AND U.S. INNOVATION 5–6 (2013), https://obamawhitehouse.archives.gov/sites/default/files/docs/patent_report.

Congress, the General Accounting Office ("GAO") found that lawsuits involving software–related patents accounted for 89% of the increase in defendants from 2007–2011, and that two–thirds of defendants during that time period were sued over software–related patents.[22] Studies have found that the majority of the patents held by NPEs were bought in the marketplace from operating companies.[23] These transfers support not only the transfer of technology but also the transfer of the legal right to sue, from operating companies that are limited in their ability to sue, due to reputational and counter–assertion risks, to those without such limits.[24] Following a number of legal developments as detailed in Section IV.A, the number of patent suits has declined recently from its peak in 2013.[25] Yet how this development has impacted the market for patents has received scant scholarly attention.

Legal academics have written dozens of studies on the topic of patent litigation by patent assertion entities alone,[26] many of them involving

pdf (showing the rise in patent litigation from 2009 to 2012 involving patents acquired from defunct companies).

22. U.S. GOV'T ACCOUNTABILITY OFFICE, GAO-13-465, INTELLECTUAL PROPERTY: ASSESSING FACTORS THAT AFFECT PATENT INFRINGEMENT LITIGATION COULD HELP IMPROVE PATENT QUALITY (2013), http://www.gao.gov/products/GAO-13-465.

23. Michael Risch, *Patent Troll Myths*, 42 SETON HALL L. REV. 457, 485–88 (2012) (finding based on studying 347 patents that 243 were initially assigned to a company, and "more than 75% of these companies were corporations while the remainder were LLCs and limited partnerships").

24. *See, e.g.*, Daniel Rubinfeld, *IP Privateering in the Markets for Desktop and Mobile Operating Systems*, 33 BERKELEY TECH. L.J. (forthcoming 2018) (describing economic incentives to assign enforcement of patent rights to NPEs).

25. Amanda Ciccatelli, *Patent Litigation Continues Sharp Downturn & Grants Bounce Back*, INSIDECOUNSEL (June 21, 2017), http://www.insidecounsel.com/2017/06/21/patent-litigation-continues-sharp-downturn-grants?slreturn=1507952354 (describing the decline in patent cases from 2013 to mid–2017 due to legal developments that include the Supreme Court's *Alice* decision and changes introduced by the America Invents Act).

26. See the studies cited by two letters sent to members of Congress in 2015: Letter from Forty Economists and Law Professors to House and Senate Judiciary Committees (Mar. 10, 2015), http://cpip.gmu.edu/wp-content/uploads/2015/03/Economists-Law-Profs-Letter-re-Patent-Reform.pdf; Letter from Fifty–One Intellectual Property Scholars to the Members of Congress (Mar. 3, 2015), http://patentlyo.com/patent/2015/03/rewards-effective-reform.html; and, the studies cited by Council of Economic Advisors in an Issue Paper entitled *The Patent Litigation Landscape: Recent Research and Developments* (March 2016), https://obamawhitehouse.archives.gov/sites/default/files/page/files/201603_patent_litigation_issue_brief_cea.pdf.

software inventions.[27] But with a few notable exceptions,[28] relatively little empirical attention has been devoted to the transactional events in a patent's life—such as the collateralization, sale, and licensing of patents.[29] The two are related, of course, with litigation often resulting from failed licensing attempts, and licenses often signed when cases are settled.

The gap in the literature is understandable in light of the lack of public information about the marketplace for patents. There is no requirement to publicly record patent licenses or sales.[30] When transactions are disclosed during the course of litigation, which are public proceedings, their terms are often kept secret behind protective orders.

But the gap is also highly problematic insofar as it produces at best an incomplete and at worst a distorted understanding of the relationship between patents and software innovation. Claims about the patent marketplace, for example extolling its virtues[31] or questioning its social utility,[32] tend not to be grounded in empirical evidence. Patent litigation involves an estimated 1–2% of all patents, yet it occupies a much larger share of policy and academic attention, creating at least two additional risks. First, neglect of commercially important but non–litigated patents may be leading to missed opportunities to observe and improve innovation and patent policy. Second, policymaking intended to address the 1–2% of litigated patents may have unintended and potentially negative

27. *See, e.g.*, Colleen V. Chien & Edward Reines, *Why Technology Customers Are Being Sued En Masse for Patent Infringement and What Can Be Done*, 49 WAKE FOREST L. REV. 235 (describing the assertion of patents against large numbers of end–user defendants based on digital innovations).

28. Two are Stuart J.H. Graham et al., *High Technology Entrepreneurs and the Patent System: Results of the 2008 Berkeley Patent Survey*, 24 BERKELEY TECH. L.J. 1255 (2009), which probed patent licensing and financing in depth by surveying entrepreneurs; and, Colleen V. Chien, *Predicting Patent Litigation*, 90 TEX. L. REV. 283 (2011), an empirical study of securitization, reassignment, and other characteristics of patents "acquired" after issuance, as well as those developed before issuance and their influence on a patent's propensity to be litigated.

29. There are a greater number of economics studies on these topics, as recounted in greater detail in Section II.B.2.

30. *See, e.g.*, Carlos C. Serrano, *The Dynamics of the Transfer and Renewal of Patents*, 41 RAND J. ECON. 686, 690 (2010) (describing the lack of a requirement to publicly record patent licenses, and providing a summary of the anecdotal data that is available).

31. Robin Feldman & Mark A. Lemley, *Do Patent Licensing Demands Mean Innovation?*, 101 IOWA L. REV. 137 (2015).

32. Michael J. Burstein, *Patent Markets: A Framework for Evaluation*, 47 ARIZ. ST. L.J. 507 (2015).

consequences for the patent system's important functions of facilitating financing, transactions, and the freedom to operate.

This Article is part of a larger project to address the substantial void in our understanding of the market for patents and patented innovations,[33] which, for the reasons elaborated in Part II, have long been considered unexplored territory. This Article leverages two datasets to address the questions that to date have been largely unanswerable in any systematic way about the role of the patent marketplace in promoting or hindering innovation. The first database catalogues "patent transfers" and includes the universe of standalone software and related patent reassignments[34] recorded at the USPTO from 2012 through 2015, as provided by Innography. The second database comprises "material technology licenses" recorded with the Securities and Exchange Commission (SEC) from 2000 to 2015. While each dataset has its strengths and limitations, discussed in depth in Part III, it should be noted that the material technology license database by its own terms has a much narrower range that does not include licenses between private companies, or agreements signed by public companies that do not reach the threshold of "materiality" that triggers disclosure.[35] For this reason, the findings reported here should be understood as reflective of a cross–section of material licenses, rather than representative of licensing in general.

This study makes several findings about the market for software innovation and its role in encouraging innovation. First, while most of the academic and policy attention devoted to software patents has focused on their litigation, this study finds, consistent with others, that the chance of a software patent being traded or licensed is much greater than the chances of it being litigated.[36] While patent litigation involves an estimated 1–2% of all patents,[37] software patents are being sold in standalone transactions at a

33. This current literature and resources are described in Section II.B.2.

34. The phrase "patent reassignments" references assignments subsequent to the initial assignment.

35. *See infra* Section III.B.2. (describing the materiality requirement).

36. *See* Serrano, *supra* note 30 (finding that about 13.5% of patents are transferred at least once over their lifetime); Stuart J.H. Graham et al., *Patent Transactions in the Marketplace: Lessons from the USPTO Patent Assignment Dataset* (Georgia Tech. Scheller College of Bus., Research Paper No. 29, 2016), http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2696147 (reporting an annual patent "churn" rate of 4.5% per year). While comparisons between studies are imperfect because each study uses a different methodology to track reassignments, all studies consistently report a greater likelihood of transfer than litigation.

37. Lerner et al. document the litigation hazard rate for a selected group of patents at about 1.29% with financial services patents almost twice as likely to be litigated. Josh

much higher rate—around 1.5–2% per *year* from 2012–2015,[38]which, assuming a patent stays in force about ten years on average,[39] leads to a transfer hazard of 15–20%. In addition, neither the decline in the enforceability of software patents over the past few years, nor the decline in levels of patent litigation generally, has not led to a corresponding slowdown in patent sales. To the contrary, this study finds that the number and share of software patents transferred actually increased between 2012 and 2015, and related work supports this observation through the end of 2016.[40] This rise may be due to bargain shopping (as prices per patent have declined), the robustness of defensive patenting strategies, and the underlying significance and importance of software innovation.

Second, this Article uses the data to probe the extent to which the market for software patents is primarily in support of the transfer of technology or the transfer of rights, with mixed results. Recent studies suggest that patent licenses rarely are accompanied by technology transfer when initiated by the patent holder.[41] But this Article's analysis of material software technology licenses reported to the SEC finds that in most cases, when patents were licensed, so were know–how, trade secrets or code. This suggest that, among this subset of licenses at least, agreements supported the transfer of technology, rather than just transferring naked patent rights.

When looking at recorded transfers of software patents from 2012–2015, however, it appears that patents are being transferred to support the transfer of technology as well as to head off or avoid disputes, or to bolster a firm's freedom to operate. Among companies for whom age information could be found, this study found software patents overwhelmingly more likely to be sold from older to younger companies, and from companies with more revenue to companies with less revenue.

Taken together, these findings support a narrative about software patents that stands in contrast to the depiction of software patents as a drag

---

Lerner et al., *Financial Patent Quality: Finance Patents After* State Street (Harvard Bus. Sch., Working Paper No. 16-068, 2015), http://www.hbs.edu/faculty/Publication%20Files/16-068_702dabb8-70c5-4917-a257-75dc8b0c4f6b.pdf. However, this study likely understates the total because of the age of the patents studied. *Id.*

38. *See infra* Section IV.A.1 fig.2.

39. *See* Dennis Crouch, *Maintenance Fees 2015*, PATENTLY-O (July 21, 2015), https://patentlyo.com/patent/2015/07/maintenance-fees-2015.html (showing the distribution of maintenance fee payments over time; and that about 60% of patents are maintained through the second maintenance fee for a lifespan of 11.5 years).

40. Brian J. Love, Kent Richardson, Erik Oliver & Michael Costa, An Empirical Look at the "Brokered" Market for Patents 33 (Sept. 16, 2017) (unpublished manuscript) (on file with author) (reporting, based on a study of brokered patent transactions, rises in the size and number of sold software patent packages from 2012–2016).

41. Feldman & Lemley, *supra* note 31.

on innovation. Software patents are much more likely to be traded or sold than litigated. Many of these transactions are happening in the shadow of competition, not only litigation, to support technology transfer and freedom to operate. In these technology transactions, software patents operate as a currency of innovation, enabling the exchange of technology and rights for money.

The Article proceeds as follows: Part II describes the theory and available evidence about the licensing and sale of patents—in particular software patents—and the role of patent transactions in supporting software innovation. Part III describes the methods, data sources, and approaches this Article uses to advance current understanding. Part IV discusses the findings and their implications. Part V concludes.

II. THEORY AND EVIDENCE REGARDING THE LICENSING AND SALE OF PATENTS

If a patent provides a right to exclude, why would patentees use them to include others through licensing or sale? This Part explores why and how technology and legal rights are transferred through patent transactions, as well as what is known about their prevalence, frequency, and role in stimulating innovation.

A. TRANSFERRING RIGHTS AND TRANSFERRING TECHNOLOGY IN THE PATENT MARKETPLACE

The purpose of the patent system, as enshrined in the Constitution, is to "promote the Progress of . . . useful Arts, by securing for limited Times to . . . Inventors the exclusive Right to their . . . Discoveries . . . ."[42] According to the "incentive to invent" story, an inventor comes up with a product, obtains a patent over it, and uses the patent to deter others from copying.[43]

42. U.S. CONST. art. I, § 8, cl. 8.

43. *See, e.g.*, Burstein, *supra* note 32, at 516. Across surveys, deterring copying is consistently reported as the top reason that inventors patent. *See, e.g.*, Graham et al., *supra* note 28; Wesley M. Cohen et al., *Protecting Their Intellectual Assets: Appropriability Conditions and Why U.S. Manufacturing Firms Patent (Or Not)* figs. 7 & 8, (Nat'l Bureau of Econ. Research, Working Paper No. 7552, 2000) (showing that 96% of the 1,478 R&D managers surveyed by Cohen and his colleagues indicated that preventing copying motivated the acquisition of their last product innovation patent); Sadao Nagaoka & John P. Walsh, *Commercialization and Other Uses of Patents in Japan and the U.S.: Major Findings from the RIETI-Georgia Tech Inventor Survey* 44 fig.13 (Georgia Tech. Sch. of Pub. Policy, Working Paper No. 47, 2009), https://smartech.gatech.edu/handle/1853/27800 (describing the results of a survey of inventors of "triadic patents"—patents

Ex ante, the inventor is encouraged to take greater risks and engage in more R&D because of the protection the patent provides; and ex post, the inventor is incentivized to make greater investments in commercialization and dissemination.[44]

Transactional justifications for the patent system adjust this story in a few ways. Ex ante, transactional freedom strengthens the basic incentive to invent as the ability of patentees to sell their technology to those who can more efficiently develop and commercialize technology "prospects"[45] raises the likelihood of a favorable return on investment. Ex post, patents make transactions more likely in several ways. First, they create defined property rights that are, unlike unregistered rights such as trade secrets, observable. The boundaries of patent rights are also more readily ascertainable than trade secrets, defining the duration of the right and the scope of the claims so that the parties do not have to do so.[46] Patents increase the confidence of patent holders in that their inventions will not be copied based on negotiation disclosures, thereby overcoming the challenge of selling information known as the "Arrow information paradox."[47] Patents

whose applications were filed in both the Japanese Patent Office and the European Patent Office and granted in the United States Patent Office—and finding that 82% of the 7,933 American inventors selected enhancing exclusive exploitation, followed by blocking, as the top answer to the question of what motivated their patenting); Gaetan de Rassenfosse & Dominique Guellec, Motivations to Patent: Empirical Evidence From an International Survey 98 tbl.2 (2008) (unpublished manuscript), www.pucsp.br/icim/ingles/downloads/pdf_proceedings_2008/08.pdf (reporting that "to prevent imitations by competitors" was the top motivator for getting patents among 604 respondents to a survey sent to randomly selected applicants of European Patent Office (EPO) patents).

44. *See* Mark Lemley, *Ex Ante Versus Ex Post Justifications for Intellectual Property*, 71 U. CHI. L. REV. 129 (2004).

45. *See, e.g.*, Ted Sichelman, *Commercializing Patents*, 62 STAN. L. REV. 341, 373–76 (2010); Michael Abramowicz, *The Danger of Underdeveloped Patent Prospects*, 92 CORNELL L. REV. 1065, 1068–70 (2007).

46. On the transactional advantages of patents over trade secrets, which are available even in the absence of compelling evidence of their impact on incentives to invent, and which do not risk destruction upon disclosure, see WILLIAM M. LANDES & RICHARD A. POSNER, THE ECONOMIC STRUCTURE OF INTELLECTUAL PROPERTY LAW (2003).

47. Robert Merges, *Intellectual Property Rights and Bargaining Breakdown: The Case of Blocking Patents*, 62 TENN. L. REV. 75 (1994) ("To sell, one must disclose the information, but once the information is disclosed, the recipient has it and need not buy it. On the other hand, if one does not disclose anything the buyer has no idea what is for sale.").

can also promote freedom to operate[48] and access to capital and talent[49] by signaling a small or young firm's innovative potential to investors[50] or banks (through the securitization process)[51] or directly, through sales or licensing.

But just as patent transfers can exploit comparative advantages in commercialization, they can also exploit comparative advantages in enforcement.[52] While both forms of transfer can promote innovation, how and whether they do on balance varies. As Justice Kennedy has noted, there is a difference between the use of patents "as a basis for producing and selling goods" and as a "bargaining tool to charge exorbitant fees."[53] Many commentators and policymakers have made similar distinctions, generally agreeing that while patent transfers that support technology transfer increase social welfare, licenses driven primarily by avoiding the cost of litigation or switching costs, rather than the value of the technology,[54] on balance decrease social welfare.[55]

---

48. *See, e.g.*, Colleen V. Chien, *From Arms Race to Marketplace: The Complex Patent Ecosystem and Its Implications for the Patent System*, 62 HASTINGS L.J. 297, 326, 328 (2010) (describing how pursuit of freedom to operate and other defensive motives contribute to patenting trends).

49. *See, e.g.*, Robert P. Merges, *A Transactional View of Property Rights*, 20 BERKELEY TECH. L.J. 1477 (2005).

50. Carolin Haeussler et al., *To Be Financed or Not… - The Role of Patents for Venture Capital Financing* (ZEW - Centre for European Econ. Research, Discussion Paper No. 09-003, 2013), http://ftp.zew.de/pub/zew-docs/dp/dp09003.pdf.

51. Aleksander Nikolic, *Securitization of Patents and Its Continued Viability in Light of the Current Economic Conditions*, 19 ALB. L.J. SCI. & TECH. 393 (2009).

52. Alberto Galasso et al., *Trading and Enforcing Patent Rights*, 44 RAND J. ECON. 275, 302 (2013) ("Our estimates suggest that patents with low values of P (defined as an estimate of probability of not having changed ownership) are more likely to be involved in transactions driven by product market gains, and patents with high P are more likely to be involved in transactions driven by enforcement gains.").

53. eBay Inc. v. MercExchange, L.L.C., 547 U.S. 388, 396 (2006) (Kennedy, J., concurring).

54. Acknowledging that it may be difficult to develop a consensus regarding whether or not a license falls into this category. *See* Colleen V. Chien, *Holding Up and Holding Out*, 21 MICH. TELECOMM. & TECH. L. REV. 1 (2014) (describing how even nuisance settlements can also function as last resorts for patentees confronted by infringers who refuse to provide license fees or "hold–out").

55. *See, e.g.*, Chien, *supra* note 16, at 342 (describing nuisance fee–driven patent litigation and settlement); FED. TRADE COMM'N, THE EVOLVING IP MARKETPLACE: ALIGNING PATENT NOTICE AND REMEDIES WITH COMPETITION (2011), www.ftc.gov/sites/default/files/documents/reports/evolving-ip-marketplace-aligning-patent-notice-and-remedies-competition-report-federal-trade/110307patentreport.pdf [hereinafter FTC REPORT]; Burstein, *supra* note 32; Robert P. Merges, *The Trouble with Trolls: Innovation, Rent-Seeking, and Patent Law Reform*, 24 BERKELEY TECH. L.J. 1583, 1588 (2009)

B. SOFTWARE PATENTS AND THE PATENT MARKETPLACE

To what extent do theories of the patent system described above explain the present relationship between software patents and software innovation? In many respects, the fit between the primary, "incentive to invent" story of the patent system and software innovation is poor.[56] Software innovations tend to be incremental, conceptual, and algorithmic; patents are supposed to be reserved for only non–obvious,[57] non–abstract, and non–mathematical inventions.[58] As property rights, patents function best when they articulate clear boundaries for the range of excluded behavior. However, software patent boundaries are notoriously "fuzzy,"[59] given their functional nature, reliance on non–specific language[60] that captures the function rather than the form of the underlying code, and the use of "patentese"[61]—the special, technical, legal language of patents.[62] Software cycles tend to be short, while patent cycles are long. As of July 2017, it took an average of seventeen months for the U.S. Patent and Trademark Office ("USPTO") to begin examining a patent application, and another 10 months for it to complete examination.[63] Under the normal default, a patent application will

(describing "inefficient, socially wasteful patent transactions" carried out by patent "trolls").

56. *See* Ronald Mann, *Do Patents Facilitate Financing in the Software Industry?*, 83 TEX. L. REV. 961 (2005) (describing advantages and disadvantages of patents for software startups based on approximately sixty interviews with software developers, venture capitalists, angel investors, banks that lend to software startups, large software and hardware firms, and others).

57. 35 U.S.C. § 103 (2012) (restricting patentability to nonobvious subject matter).

58. Bilski v. Kappos, 561 U.S. 593 (2010) (stating that "abstract ideas," "mathematical formula[s]," and "algorithms" are not patentable).

59. JAMES BESSEN & MICHAEL J. MEURER, PATENT FAILURE: HOW JUDGES, BUREAUCRATS, AND LAWYERS PUT INNOVATORS AT RISK 10 (2008).

60. To take one recent example, does the term "distributed learning control module" cover any software or hardware that carries out a set of basic functions, specifically, the functions of "receiving communications transmitted between the presenter and the audience member computer systems and for relaying the communications to an intended receiving computer system and for coordinating the operation of the streaming data module"? US Patent No. 6,155,840 (filed Sept. 18, 1998). Until recently, even the courts have not been sure. *See, e.g.*, Williamson v. Citrix Online, LLC, 792 F.3d 1339 (Fed. Cir. 2015). The use of vague terms in software patents like "module," has prompted one parody patent drawing that consists of a combination of "thing-a-ma-jigs," "stuff," "whatzits," "doo-hickies," and "you know." FLICKR (2011), https://www.flickr.com/photos/opensourceway/6554315093/sizes/l.

61. Sean B. Seymore, *The Teaching Function of Patents*, 85 NOTRE DAME L. REV. 621, 627–33 (2010).

62. *Id.* at 633–34.

63. *Data Visualization Center,* U.S. PATENT & TRADEMARK OFFICE, http://www.uspto.gov/dashboards/patents/main.dashxml (last visited Oct. 8, 2017).

publish at eighteen months,[64] and a patent can stay in force for up to twenty years from the date of filing. But in fields like smartphone mobile applications (or "apps"), the market environment is changing quickly.[65] Many apps fail within weeks if not months, making it hard to know ex ante whether or not the software is worth protecting.[66] Imitation cycles are also short, with the most successful applications imitated within months;[67] meaning that the whole cycle from conception of a feature for the mobile app, to its copying by another can happen even before a patent application matures into a patent.

According to a recent study by Christian Helmers and his colleagues, only a tiny share—around 0.04%—of smartphone applications available in the Apple iOS store are protected by app–relevant patents.[68] There are obviously counterexamples to the app industry—software areas that are heavily patented, and rely on much longer product cycles. Even in the app environment, patented apps command higher prices, and are more likely to be rated extensively.[69] But the sense that software is different[70] has led prominent leaders in the industry to reject the premise that software patents are necessary to incentivize software innovation.[71] As the 2008 Berkeley Patent Survey found, two–thirds of software entrepreneurs do not have or seek patents.[72]

---

64. 35 U.S.C. § 122 (2012).

65. Sebastian G. J. Brandes Kraaijenzank, Innovation without Patents? Evidence from the Smartphone App Markets (June 2013) (unpublished M.B.A. thesis, Universidad Carlos III de Madrid) (on file with author).

66. This assumes, of course, that the apps contain protectable inventions.

67. *Id.* at 24–27 figs. 2–5.

68. *Id.* at 17 tbl. 4. Across all app stores in the study, the protection rate is 4.5%.

69. *Id.* at 19 tbl. 5.

70. *See, e.g.*, Github Conversation Between Marc Andreesen and Peter Thiel, https://gist.github.com/jm3/2669267 (last visited Oct. 8, 2017) ("There are some areas in tech—drugs and mechanical equipment, for instance—where parents are fundamental. In these areas there are long established historical norms for who gets to do what. But in software, things change extremely quickly. The big companies used to have huge war chests full of patents and use them to squash little guys. Now they're fighting each other. The ultimate terminal state of big companies seems to be a state in which they build nothing. Instead, they just add 10,000 patents to their portfolio every year and try to extract money through licensing. It'd be nice if none of this were the case. But it's not startups' fault that the patent system is broken. So if you have a startup, you just have to fight through it. Find the best middle ground strategy.").

71. *See, e.g.*, Fred Wilson, *Enough Is Enough*, BUS. INSIDER (June 1, 2011), http://www.businessinsider.com/enough-is-enough-2011-6 ("I believe that software patents should not exist.").

72. Graham et al., *supra* note 28, at 1277 tbl.1.

But the same Survey found that among venture backed software startups, the majority had patents.[73] One of the reasons that venture capitalists like patents is because they can distinguish firms with unique, proprietary technologies, and provide salvageable assets should the firm fail. Within firms, the successful pursuit of patents can support the creation of jobs and sales growth.[74] But filing for patents takes resources away from engineering tasks,[75] and patent litigation demands are a distraction and strain on the innovative enterprise, sometimes taking a significant operational toll on small companies.[76]

While valuable, studies about filing for, obtaining, and litigating patents are at the periphery of the patent market. Patent licenses signed as the result of patent litigation are a highly selected part of the patent market, and because they are formed ex post, they also tend to take place after technology has been transferred or copied, or independently invented.[77] Funding events that follow the issuance of patents do not represent market transactions of the patent, and it is hard to tease apart the extent to which patent–holding causes funding events, rather than being a characteristic of fundable, well–run startups. Studies that focus on the strategic acquisition of patents in order to litigate them,[78] in turn, do not address sale of patents for commercialization and other objectives.

The present study is different, because it directly observes actual transactions—licenses and sales—in the marketplace for patented software

73. *Id.*

74. *See, e.g.*, Joan Farre-Mensa et al., *The Bright Side of Patents* (U.S. Patent & Trademark Office, Economic Working Paper No. 2015-5, 2016), http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2704028; David H. Hsu & Rosemarie H. Ziedonis, *Patents as Quality Signals for Entrepreneurial Ventures*, 2008 ACAD. MGMT. PROC. 1 (2008) (finding that patents have a positive effect on startup company value).

75. Ronald Mann, *Do Patents Facilitate Financing in the Software Industry?*, 83 TEX. L. REV. 961, 982–84 (2005).

76. *See, e.g.*, Colleen V. Chien, *Startups and Patent Trolls*, 17 STAN. TECH. L. REV. 461 (2014); Colleen V. Chien, *Patent Assertion and Startup Innovation* (New America Foundation, Open Technology Institute White Paper, 2013), http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2321340; James Bessen, *The Evidence Is In: Patent Trolls Do Hurt Innovation*, HARV. BUS. REV. (Nov. 2014), https://hbr.org/2014/07/the-evidence-is-in-patent-trolls-do-hurt-innovation/; *see also* Letter from Startup Investors to Congress (Mar. 17, 2015), http://engine.is/wp-content/uploads/VCsforPatentReformLtr2015-1.pdf.

77. Christopher A. Cotropia & Mark A. Lemley, *Copying in Patent Law*, 87 N.C. L. REV. 1421, 1465 (2009).

78. *See, e.g.*, Fiona M. Scott Morton & Carl Shapiro, *Strategic Patent Acquisitions*, 79 ANTITRUST L.J. 463 (2014).

innovations.[79] By studying recorded sales in general, and reported, material licenses in particular, these transactions span a variety of reasons that patents are licensed and sold, enabling their direct comparison.

1. Transfers of Rights vs. Transfers of Technology

This Article distinguishes between patent transactions that transfer technology and patent transactions that transfer rights or liability. A patent–centric view glosses over this distinction, finding that all patent transactions happen in the shadow of litigation, and are driven by consideration of how a court might later view the settlement.[80] But while some licenses are motivated by the desire to avoid suit, others are motivated by the desire to gain technology. Rather than happening in the shadow of litigation, agreements to transfer the technology happen in the shadow of the market and competition; for example, in the race to be first to market. Rather than being driven by the cost of litigation, the price of licenses to transfer technology is driven by the value of the technology and the extent to which the technology can accelerate development of a product or yield a return for the business. Those forced to take patent licenses in order to avoid being sued are reluctant licensees, those who seek out licensing partners in order to access their technology represent willing licensees.

The distinction has not only descriptive but also normative significance. Those who extol the virtue of patent markets credit to them the benefits of the technology transfer, including gains associated with specialization in innovation. But not every patent license achieves these gains. Some transfers of rights are in effect just preemptive legal settlements that eliminate the risk of potentially rent–seeking lawsuits. While such transfers could be welfare–enhancing, insofar as they support the exclusion that animates the incentive to invent story,[81] they can also be welfare–reducing when they involve the enforcement of a wrongly–issued patent, or encourage enforcement and settlements based on the cost of litigation and

79. *Cf.* Brian W. Carver, *Share and Share Alike: Understanding and Enforcing Open Source and Free Software Licenses*, 20 BERKELEY TECH. L.J. 443, 448–50 (2005) (analyzing open–source licensing agreements under the GNU General Public License); Christian Chessman, *A "Source" of Error: Computer Code, Criminal Defendants, and the Constitution*, 105 CALIF. L. REV. 179, 223, 226 n.348 (2017) (discussing the role of open–source innovation in the proprietary development of software).

80. Jonathan S. Masur, *The Use and Misuse of Patent Licenses*, 110 NW. U. L. REV. 115 (2015).

81. For example, defensive patent aggregators like RPX may buy a patent in order to remove the threat of litigation from its member companies.

switching costs, rather than the value of the technology.[82] The following paragraphs review existing work as a backdrop for the present study.

2. Existing Studies of the Patent Marketplace

Lamoreaux and Sokoloff have performed the most significant early work on markets for technology in the nineteenth century using the patent record.[83] Made known by weekly descriptions published in The Scientific American starting in 1845 and the patent lawyers and agents who acted as intermediaries, nineteenth century patents frequently changed hands.[84] Lamoreaux and Sokoloff estimate that approximately 12% to 28% of patents were assigned more than once, including through corporate acquisition.[85] These sales, as well as other information, provide evidence that patents supported the buying and selling of technology more broadly, not just the buying and selling of the patents themselves. But other studies have documented the use of nineteenth century patents for the purpose of transferring the rights to sue others as well, in the context of farming and railroad patents.[86] In the case of farming patents, trivial improvements formed the basis of patents that were used to demand royalties from unsuspecting farmers, many of whom bought the allegedly infringing technology.[87]

Though these transactions predated the rise of digital technology, Serrano's study of patent reassignments from 1980 to 2001 specifically considered the prevalence of patent transfers among different industries. He found that patents in the computer and communications as well as the drug and medical industries had the highest likelihood of being transferred during their lifetime, about 13.5%.[88] In 2015, the USPTO's Chief Economist Office released the "USPTO Patent Assignment Dataset," a database

82. Some might argue that even such transfers as these may have positive welfare effects, insofar as liability transfers reduce the need for litigation, and a patent, even if wrongfully issued, induces socially valuable racing.

83. Naomi R. Lamoreaux & Kenneth L. Sokoloff, *Inventors, Firms, and the Market for Technology in the Late Nineteenth and Early Twentieth Centuries* (Nat'l Bureau of Econ. Research, Historical Working Paper No. h0098, 1997), http://www.nber.org/papers/h0098.pdf.

84. *Id.* at 22–24.

85. *Id.* at 52 tbl.1.6.

86. *See* Chien, *supra* note 16 (discussing the parallels between the historical and modern patent controversies); Christopher Beauchamp, *The First Patent Litigation Explosion*, 125 YALE L.J. 848 (2016) (offering an overview of these chapters in the history of the agrarian and railroad industries).

87. *See* Earl W. Hayter, *The Patent System and Agrarian Discontent, 1875-1888*, 34 MISS. VALLEY HIST. REV. 59, 61 (1947).

88. Serrano, *supra* note 30, at 686.

covering approximately six million assignments and other transactions recorded from 1970 to 2014.[89] According to these records, recent patents[90] are more likely to be transferred than patents from earlier decades, the growth led in particular by the transfer of patents in the computer and communications sectors.[91] Graham and his co–authors find, based on analyzing this data, a yearly churn rate of 4.5% in 2014, as compared to Serrano's lifetime transfer rate of 13.5%. However, differences in the methodology between Graham et al. and Serrano probably explain the discrepancy between these numbers.

Because these studies were based solely on patent records, neither probed the motives for or conditions of patent transfers. However, a pair of studies have looked specifically at the relationship between transfer and litigation. While both studies find, on average, that the transfer of patents reduces litigation risk,[92] Galasso and his coauthors also find that patents traded to smaller entities were associated with a greater chance of litigation.[93] Sales from larger companies to smaller NPEs[94] fit this trend.

In contrast with data about patent sales, which are routinely publicly recorded, public data about patent licenses is scarce.[95] There are no requirements to record patent licenses, which are regarded as highly sensitive[96] even when they involve publicly funded patents.[97] Surveys

89. Graham et al., *supra* note 36, at 2.

90. The records specifically deal with patents issued since 2000–2005.

91. Graham et al., *supra* note 36, at 17.

92. Chien, *supra* note 27; Galasso et al., *supra* note 52.

93. Galasso et al., *supra* note 52, at 34.

94. Michael Risch, *Patent Troll Myths*, 42 SETON HALL L. REV. 457, 485–88 (2012) (finding, based on studying 347 patents, that 243 were initially assigned to a company, and "more than 75% of these companies were corporations while the remainder were LLCs and limited partnerships").

95. *See e.g.* Iain M. Cockburn, Is the Market for Technology Working? Obstacles to Licensing Inventions, and Ways to Reduce Them (June 8, 2007) (unpublished manuscript), https://www.researchgate.net/publication/267839147_Is_the_Market_for_Technology_Working_Obstacles_to_Licensing_Inventions_and_Ways_to_Reduce_Them.

96. As a result, studies generally rely on proprietary databases. *See, e.g.*, Bharat N. Anand & Tarun Khanna, *The Structure of Licensing Contracts*, 48 J. INDUS. ECON. 103, 105–06 (2000) (analyzing 1,612 patents from the Strategic Alliance database of Securities Data Company); Joshua S. Gans et al., *The Impact of Uncertain Intellectual Property Rights on the Market for Ideas: Evidence from Patent Grant Delays*, 54 MGMT. SCI. 982 (2008) (analyzing a sample of 200 licenses announced between 1990 and 1999 in the Security Data Corporation database).

97. Arti K. Rai & Bhaven N. Sampat, *Accountability in Patenting of Federally Funded Research*, 30 NATURE BIOTECHNOLOGY 953 (2012).

estimate that about ten percent of patents are licensed,[98] but that the extent of licensing depends on the entity size.[99] The few empirical studies of licensing that do exist, generally conducted by economists, focus on the prices[100] and strategies behind licensing.[101]

3. Patent–Only–Licenses vs. Licenses for Know–How

One proxy for whether patent licensing supports technology transfers or liability transfers is the extent to which licenses provide only patent rights as opposed to patent rights with know–how. Patent licenses that include knowledge, know–how, personnel, or joint venture relationships are more likely to represent direct transfers of technology, whereas the transfer of "naked" patent rights is more likely to primarily represent a transfer of liability between the parties. Which type of patent license is more prevalent? The answer varies considerably based on context. Varner's study of 1,458 patent licenses including patent assignments, which were attached as exhibits in filings to the SEC, found that 56% of patent agreements included know–how, while 33% were "bare patent" transfers and 11% were patent assignments,[102] consistent with earlier and smaller samples.[103] These proportions were roughly consistent across the industries he considered,

98. Dietmar Harhoff, *The Role of Patents and Licenses in Securing External Finance for Innovation*, 14 EIB PAPERS 74, 81 (2009) (summarizing surveys by Motohashi (2008), Nagaoka and Kwon (2006), and Gambardella et al. (2007)).

99. Paola Giuri et al., *Everything You Always Wanted To Know About Inventors (But Never Asked): Evidence From the Patval-EU Survey* (Munich Sch. Mgmt., Univ. Munich, Discussion Paper No. 2006-11, 2006), https://epub.ub.uni-muenchen.de/1261/1/LMU_WP_2006_11.pdf.

100. *See, e.g.*, GREGORY J. BATTERSBY & CHARLES GRIMES, LICENSING ROYALTY RATES (2017 ed.); Deepak Hedge, Essays on Institutions and Innovation (2010) (unpublished Ph.D. dissertation, University of California, Berkeley), http://escholarship.org/uc/item/0sp3n4sk; Jonathan E. Kemmerer & Jiaqing Lu, *Profitability and Royalty Rates Across Industries: Some Preliminary Evidence*, KPMG GLOB. VALUATION INST. (Nov. 19, 2012), https://assets.kpmg.com/content/dam/kpmg/pdf/2015/09/gvi-profitability.pdf.

101. *See, e.g.*, Goretti Cabaleiro Cerviño, Firm Strategies Behind the Establishment of Licensing Agreements (Apr. 2014) (unpublished Ph.D. dissertation, University of Madrid), http://e-archivo.uc3m.es/bitstream/handle/10016/18988/goretti_cabaleiro_tesis.pdf.

102. Thomas R. Varner, *An Economic Perspective on Patent Licensing Structure and Provisions*, 47 LES NOUVELLES 28, 31 (2012).

103. Victor Braun, *Licenses as Critical Sources of Innovation*, 43 LES NOUVELLES 225, 226 (2008) ("Contractor (1985) found that in the early 1980s 75 percent of U.S. license agreements contained know-how transfers. Vickery (1988) in a Les survey of 119 international licensing transactions detected 67 percent. In the Chemical Industry, all but the simplest licenses involve a mixture of patents and know-how.").

including "high–tech."[104] But when Feldman and Lemley surveyed those who had received licensing demands, they found the opposite: that in the overwhelming majority of cases, the subsequent license was *not* accompanied by the transfer of knowledge, know–how, personnel, joint venture relationships, or other indicia of technology transfer.[105] Like Varner's study, the Berkeley Patent Survey presents a mixed view, based on surveying over 1,300 startups in mid–2000. Among venture–backed software startups, 12% licensed in technology.[106] About 70% of them did so to gain knowledge, technology, or know–how while approximately a quarter of firms did so only to avoid a dispute, and *not* to gain technology.[107] A quarter of software startups, and 67% of venture–backed startups overall had patents.[108]

4. Exclusivity Provisions

Another way to distinguish between licenses that transfer technology and those that transfer liability is to look at the exclusivity provisions. An exclusive license enables the licensee, with the right to exclude conferred by the patent, to "step into the shoes" of the patent holder with the exclusive right to commercialize the invention. A cross–license, on the other hand, represents the exchange of permissions to practice the technology—one that promotes freedom to operate but, on balance, does not necessarily lead to more technology being transferred. As such, nonexclusive licenses do not transfer the incentive to commercialize provided by a patent's exclusivity.

A number of studies have looked at the level of exclusivity present in patent licenses, again with mixed results. Anand and Khanna's study of licensing deals involving at least one U.S. participant between 1990 and 1993 reported that more than 30 percent of the 1,612 deals involved exclusive licenses.[109] However, there were strong industry differences. Only 15% of "electronic" company licenses were exclusive, while over 50% of "chemical" company licenses were.[110] But electronic industry licenses (20%) were twice as likely to be cross–licenses as chemical licenses

104. Varner, *supra* note 102, at 31 tbl.1. (explaining that the "high-tech" category included: Computer Software, Computer Hardware, Electronic Components, Instrumentation, and Telecommunication firms).

105. Feldman & Lemley, *supra* note 31, at 157–71 figs. 5–28.

106. Graham et al., *supra* note 28, at 1318.

107. *Id.*

108. *Id.* at 1277.

109. Anand & Khanna, *supra* note 97, at 109.

110. *Id.* at 115 tbl. III(i).

(10%).[111] A number of studies have also found a relatively higher level of exclusive licenses among university and biotechnology patents. In their review of 1,715 patents developed at the University of California and the Department of Energy National Laboratories between 1977 and 2009, Drivas and his colleagues found that the overwhelming majority were exclusively licensed.[112] In a parallel study of university patents covering DNA published in 2006, Pressman found that exclusivity provisions varied by licensee size. The smaller the company, the more likely the license was exclusive.[113]

In sum, while existing studies of patent sales and licenses provide a glimpse of the role of patent transactions in innovation, they raise just as many questions as they answer in the context of the central issue of whether software is "eating the world" despite or because of software patents. Serrano and his colleagues have demonstrated that patent sales have been happening to a considerable degree, reducing litigation risk except when sales to larger entities are made. However, his study, which ends in 2000 transactions, predates many of the major developments in the software patent law as well as the software marketplace.[114] It also does not focus on software patents. The same is true of all of the existing studies of patent license terms. The Khanna and Anand study, which comes closest, studies licenses that are over two decades old. Given the importance of software innovation, it is worth building upon what is known by focusing specifically on software patents, software companies, and software sales and licenses. The rest of this study uses several sources to attempt to do this, with a focus on two main questions:

- How robust is the paid market for software innovation, when measured through the lens of software patent sales and software licenses?
- To what extent are the licensing and sale of software patents facilitating the transfer of technology as opposed to legal liability, based on an examination of the ways in which patents are being redistributed?

The next section outlines the methods, sources, and assumptions used, and the following section, outlines the main findings.

---

111. *Id.*

112. Kyriakos Drivas et al., Academic Patent Licenses: Roadblocks or Signposts for Nonlicensee Cumulative Innovation? 9 (Aug. 29, 2014) (unpublished manuscript), http://ssrn.com/abstract=2489231.

113. Lori Pressman et al., *The Licensing of DNA Patents by US Academic Institutions: An Empirical Survey*, 24 NATURE BIOTECHNOLOGY 31 (2006).

114. *See, e.g.*, FTC REPORT, *supra* note 55.

III. DATA SOURCES AND METHODOLOGY

To explore the market for software innovation and the role of patents in supporting this market, this study drew upon several novel sources of data. Despite the recent growth in empirical patent scholarship, law academics have generally paid less attention to markets for technology for several reasons. First, data on patent transactions has been actually or practically inaccessible or in an unusable form, for the reasons described below. In addition, patent scholars have generally paid less attention to the use of patents for commercialization, signaling, and financing purposes, which these data sources reflect, and more attention to the pursuit and litigation of patents. A focus on these "exclusionary" uses of patents is consistent with the constitutional idea of promoting the progress of science and the useful arts, by rewarding innovators through the right to exclude others from the marketplace.[115]

But recent developments have both highlighted the importance of considering the "middle layer" of patent transactions, and chipped away at obstacles to studying it. The high–profile purchases of patents by Apple and Google described earlier drew attention to the importance of patents and the freedom to operate. At the same time, the Obama Administration's commitment to "open data" and decision to treat government–generated data as public assets has led to the opening of hundreds of thousands of government datasets.[116] These datasets drive government accountability and transparency, spawn new businesses, and support existing ones.[117] Thus, though one of the two enumerated duties of the USPTO is to "be responsible for disseminating to the public information with respect to patents and trademarks,"[118] only in the last 10 years, in concert with the creation of the Office of the Chief Economist, has the agency engaged in the release of large quantities of patent data in digital form. This data identifies not only the details of patent prosecution, but ownership and other events that occur

115. *See* Justin R. Orr, *Patent Aggregation: Models, Harms, and the Limited Role of Antitrust*, 28 BERKELEY TECH. L.J. 525, 528 (2013) (analyzing the constitutional policy goals grounding the grant of patent rights).

116. *See, e.g.*, DATA.GOV, https://www.data.gov (last visited Oct. 8, 2017). These datasets pertain to everything from disaster relief, to information about Medicare and Medicaid services, to sexual assaults on campuses. *See id.*; *Case Studies of US Open Data*, PROJECT OPEN DATA, https://project-open-data.cio.gov/; *Open Data Community Events*, PROJECT OPEN DATA, https://project-open-data.cio.gov/.

117. *Id.*

118. 35 U.S.C. § 2(a)(2) (2012).

over a patent's lifetime.[119] These developments have been a boon to the more than 135 patent data companies[120] that exploit the application of machine learning and artificial intelligence techniques to code, clean, and ultimately transform raw open government data on the application, maintenance, licensing, securitization, and sale of patents, as leveraged in this analysis into useable insights. As highlighted earlier, the importance of the market for patents and technology, the range of non–exclusionary uses of patents, and our understanding of these developments has grown in recent years. Thus, in addition to the development of the "supply" of patent data, the "demand" for this data, as companies seek technology and financing partners, has also grown.

A. IDENTIFYING "SOFTWARE" AGREEMENTS AND PATENTS

In order to explore the importance of software licenses and the role of patents in supporting software innovation, I had to identify "software" companies, "software" licenses, and "software" patents, well–known to be challenging tasks. Previous researchers have developed several approaches for identifying software patents: keyword searching (i.e. for "computer program" or "software")[121] and patent classification[122] filtering (i.e. for classes G06F "Electrical Digital Data Processing" or G06F "Recognition Of Data; Presentation Of Data; Record Carriers; Handling Record Carriers").[123] To find "pure" software *companies,* Graham et al. has selected companies falling within several Standard Industrial Classification ("SIC") codes.[124] This work relies on all three approaches—keyword searching (and keyword coding) to identify software agreements, patent class codes to identify software patents, and SIC codes to identify pure software companies. Given the broad distribution of software innovation,[125] it is likely that the classification–based identification techniques used

119. Before these releases, the USPTO would provide certain data upon request but charge fees in the thousands to get it. In 2010, the USPTO, in partnership with Google, released a large amount of transactional data about patents and trademarks, including grants, assignments, and maintenance fees, publicly available for free. *See* Chien, *supra* note 27, at 300 n.110.

120. Colleen V. Chien & Brian J. Love, Comment to the USPTO on Quality Case Studies 1 (Feb. 2016), http://www.uspto.gov/sites/default/files/documents/casestudies_f_chien%26love_12feb2016.pdf (referencing these benefits).

121. James Bessen & Robert Hunt, *An Empirical Look at Software Patents*, 16 J. ECON. & MGMT. STRATEGY 157 (2007).

122. This approach is based on the CPC and IPC schemes.

123. Stuart Graham & David Mowery, *Intellectual Property Protection in the U.S. Software Industry*, *in* PATENTS IN THE KNOWLEDGE-BASED ECONOMY (Wesley M. Cohen & Stephen A. Merrill eds., 2003).

124. SIC Codes 7371, 7372, 7373, 7379. *See* Graham et al., *supra* note 28, at 1269.

125. *See* Branstetter et al., *supra* note 3.

underestimate the scope of software patents and companies in which software innovation is occurring. This Article therefore proceeds with caution, using these measures as a basis for performing and reporting *relative* trends and prevalence, rather than considering them to represent comprehensive measures of software innovation.

B. DATA SOURCES

To understand the market for software innovation through the lens of software licenses and software patent sales, the study relied primarily on two databases: the ktMINE database of material technology licenses reported to the SEC, and the Innography database of patent transfers. Though populated with open government data, each database is proprietary, introducing several limitations to this study.

First, their use precludes the release of the underlying data analyzed by this study and complicates replication efforts. Second, the databases themselves contain known coverage gaps, such as unrecorded transactions and transactions involving patent applications that were abandoned prior to publication. However, even more problematically, they may include unknown gaps or otherwise be incomplete, biasing the data in unknown ways. Third, reliance on the coding of others subjects the analysis to the risk that the coding contains errors or may be incorrectly interpreted.

I took several measures to minimize the impact of these defects. First, I describe in the Article what we know about the databases, and carried out confirmatory checks using independent coding along the way. To the degree permitted under the license agreement, I also provide information about the search approaches I used. In addition to using raw open government data, I relied upon additional codings supplied by the providers, as described in greater detail below. To avoid interpretational errors with respect to these codings, I conferred closely with each provider regarding their data sources and methodology and carried out independent confirmatory codings in a number of cases to ensure that my understanding was correct.

1. Patent Sales Data

Although there is no obligation to publicly record ownership or transfers of patent rights, doing so provides legal rights against those who might attempt to later purchase the patent.[126] However, the task of identifying what patents have been sold, to whom, and under what terms, has been complicated by the large variety of recordable "conveyances" of patent

126. Alicia Griffin Mills, *Perfecting Security Interests in IP: Avoiding the Traps*, 125 BANKING L.J. 746 (2008).

rights, including securitizations, licenses, intra–company transfers of patents, and merger and acquisition–based transfers.[127] As a result, the task of separating "true transfers" of a patent from other types of conveyances presents a significant obstacle to doing research on the patent market. About 10% of conveyances recorded at the USPTO represent true inter–company transfers.[128]

To find "true transfers" of software patents[129] that had been recorded at the USPTO between 2012 and 2015. I worked with Esmaeil Khaksari of Innography and drew upon Innography's "PMT" database, which is comprised of conveyance data that has been cleansed and processed so that only true, inter–company transfers outside of the context of the merger or acquisition are left.[130] This study found 30,898 reassignments of software patents from January 1, 2012 to December 31, 2015, some involving the same patent, together representing the transfer of 25,210 unique patents. To determine the rate at which patents were being transferred, I had to estimate the universe of possibly transferable patents. This study included any patent in force during the period of transfer in this denominator (N=433,430).[131]

2. "Significant" Software Technology Licenses

Although license data is generally not available,[132] publicly traded companies are required by SEC regulations to report in their filings,

127. Form PTO-1595, the "Recordation Form Cover Sheet" enables recordation of 8 different types of conveyances, including "Other." *See Form PTO-1595*, U.S. PATENT & TRADEMARK OFFICE (Apr. 30, 2015), http://www.uspto.gov/forms/pto1595.pdf; *see also* Chien, *supra* note 28, at 311.

128. Graham et al., *supra* note 36, at 54 fig.6; *accord* INNOGRAPHY INC., PATENT MARKET TRACKER: FALL 2015 KEY TRENDS (2016), https://www.innography.com/public/upload/files/general-files/Innography-Patent-Market-Tracker.pdf (last visited Oct. 8, 2017) (estimating the share of conveyances that are interfirm assignments to be 15%).

129. This term is used as defined by Graham & Mowery's CPC–based classification. *See* Graham & Mowery, *supra* note 123.

130. Because of the way that transfers are evaluated, the PMT excludes patent transfers that follow acquisitions of companies where the child is merged into the parent entity. However, transfers that support spin–outs or transfers to entities that are distinct from the original patent holders are still included.

131. To determine which assets were in force, this study used actual and projected expiration dates of the patent which are estimated by Innography by taking into account patent type, priority date, patent term adjustments, abandonments, and maintenance activities, but which do not include terminal disclaimers. *See Overview: Patent*, INNOGRAPHY (last visited Oct. 8, 2017), http://education.innography.com/overview-patent.

132. The lack of public data about technology licenses is a well–known impediment to research in this area. While technology and the permissions to use it are routinely exchanged in return for money or other consideration, there is no requirement that licensing transactions be publicly recorded. Even when one party might be willing to disclose what

"material definitive agreements not made in the ordinary course of business."[133] While I refer collectively in this article to these publicly filed agreements as the "SEC Database," in fact, there is no central repository of such agreements or easy way of identification in the SEC record, due to the lack of designation of such licenses and the non–standard ways in which agreements are formed and referenced by parties.[134] Although this study was able to leverage the aggregation, cleaning, and coding of these licenses by the proprietary vendor ktMINE, SEC license data has several structural limitations that are worth discussing upfront. First, in contrast to public records about patent sales, which give rise to protections against subsequent purchases of a patent by any transactor, only a small subset of agreements triggers SEC reporting requirements—agreements that are material to a public company, which in turn comprise only a small subset of all companies. As such SEC licenses are surely not representative of agreements in general,[135] but rather agreements that survive two significant filters: they are relevant to a publicly traded company, and substantial enough to be considered material. As a result, these agreements are not representative of commercial technology licenses in general but are biased towards larger rather than smaller agreements, and reported by smaller rather than larger firm

---

they paid or what they were paid, or other terms of the agreement, non–disclosure agreements typically prevent the divulgence of license details, even selectively. *See, e.g.*, Anne Kelley, *Practicing in the Patent Marketplace*, 78 U. CHI. L. REV. 115, 117 (2011); Jorge L. Contreras et al., *Study Proposal – Commercial Patent Licensing Data* (Univ. of Utah Coll. of Law, Research Paper No. 164, 2016), http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2755706.

133. U.S. SEC. & EXCH. COMM'N, OMB No. 3235-0060, Form 8-K, Item 1.01, http://www.sec.gov/about/forms/form8-k.pdf.

134. Cockburn, *supra* note 95, at 3 ("[L]icense agreements are typically complex, contingent contracts, they are difficult to value or assess, or even count up for statistical purposes. . . . Very few—if any—national statistical agencies collect comprehensive data on technology licensing activity, and the coverage, accuracy and content of databases sold by private vendors is very difficult to assess independently."). A cottage industry of companies that harvest, collect, clean, and code this data addresses this gap, including RECAP, RoyaltyStat, Biosciences Advisers, and ktMINE. *See, e.g.*, Robert Reilly, *Analyzing Intellectual Property Royalty Rate Data*, AM. INST. CERTIFIED PUB. ACCOUNTANTS (Nov. 2013), http://www.willamette.com/pubs/presentations2/reilly_aicpa_ipanalysis_nov13.pdf.

135. *See, e.g.*, Tom Varner, An Economic Perspective on Patent Licensing Structure and Provisions (2011) (unpublished manuscript) (on file with author) (comparing SEC licenses to other agreements author reviewed in the course of litigation and expert witness preparation and finding that the undisclosed agreements "include a greater percentage of cross-licenses, royalty-free licenses, and fixed fee licenses than observed in the dataset analyzed for this paper.").

This study used ktMINE's licensing database, which includes over 100,000 material agreements, collected from public sources, primarily the SEC Database. This study's analysis was performed using ktMINE's "Royalty Rate Analyzer" tool, which contains about 16,000 intellectual property license agreements with royalty terms, a subset of the total.[136] This study relied upon ktMINE's coding of basic facts about each agreement including the licensor, licensee, effective date of the license, industry of the agreement, agreement type,[137] and keywords indicating the subject matter of the license.[138]

In order to focus on agreements that cause the diffusion of technology between firms, I excluded certain types of agreements such as asset purchases (typically, associated with M&A activity) as well as marketing, distribution, and services agreements. The "technology agreements"[139] comprised about 20–25% of all agreements, and this study focused on the subset of licenses with an effective date of 2000 through 2015 (N=6,109). These effective dates were chosen in order to capture recent trends in licensing. However, due to lags between the execution and recordation of licenses, the dataset has relatively fewer licenses from recent years compared with older years.

Within this group of technology agreements, I focused on "software" technology agreements, as coded by ktMINE, yielding 1,431 licenses. I read many of these licenses to confirm that they were indeed about software and, replicating Bessen and Meurer's keyword identification approach,[140] found a roughly equivalent number of agreements (1,451). Within software technology licenses, I distinguished between agreements in which patents were mentioned (N=1,163) and those where copyrights, trade secrets, trademarks, patents, or software source code[141] were coded as "core" to the agreement by ktMINE. Based on their methodology, patents were core to 480 of the software technology agreements, which included both

136. *Royalty Rate Benchmarking Guide 2015/2016 Global Edition*, BUS. VALUATION RES. 5 (2015), http://www.bvresources.com/pdfs/RoyaltyRateGuide_2015_Excerpt.pdf.

137. *See id.*

138. *See id.*

139. I included the following agreements types in this category: cross–licenses, joint development, manufacturing/process intangible, other, and software. I excluded asset purchases, distribution, franchise, marketing intangible, and service agreements from the analysis.

140. I specifically looked to find agreements that included the term "software" or "computer program," as described by Bessen & Meurer. *See* BESSEN & MEURER, *supra* note 59.

141. For each, I worked with ktMINE to identify the relevant agreements, based on an exhaustive list of keywords covering each concept.

technology licenses and asset transfers. I worked with research assistants to code the provisions of software agreements where patent rights were also transferred outside the context of an asset transfer (N=245).

To establish a baseline from which to evaluate the prevalence of licenses, this study took several steps. The study considered the prevalence of reporting among "pure software" firms as defined by Graham and his colleagues that were eligible to report licenses over the studied period. These firms fell into three SIC codes: prepackaged software firms such as Microsoft, IBM, and Adobe Systems Inc. (SIC 7372),[142] computer integrated systems design firms like Fujitsu, and Mentor Graphics Corp. (SIC 7373),[143] and companies that provide computer programming services like Sabre Corporation or General Dynamics Information Technology (SIC 7371).[144]

Because companies are routinely listed and delisted from public exchanges, at times within the span of just a few years, taking a single year snapshot does not yield an accurate count of the universe of companies eligible to file material agreements. Therefore, this study next used COMPUSTAT to generate an aggregate list of companies within the relevant SIC codes in each of five years (2000, 2004, 2008, 2012, and 2014). Out the five–year period, there were 1,140 unique public "pure software" companies within COMPUSTAT. This study further pulled revenue from the year of the agreement to determine the prevalence of reporting among different revenue bands. For companies with reported revenue, this approach had the advantage of being available for multiple years, including the effective year of the relevant transaction, for most but not all companies.[145]

3. Company and Revenue Data

The study integrated several types of company and industry–level data into the analysis, including revenue, age of founding, and SIC code. To profile public companies, this study relied primarily on COMPUSTAT and

142. *SIC 7372 Prepackaged Software*, ADVAMEG, INC. (2016), http://www.referenceforbusiness.com/industries/Service/Prepackaged-Software.html.

143. *SIC 7373 Computer Integrated Systems Design*, ADVAMEG, INC. (2016), http://www.referenceforbusiness.com/industries/Service/Computer-Integrated-Systems-Design.html.

144. *Business List - SIC 7371 - Computer Programming Services*, SICCODE.COM, http://siccode.com/en/codes/sic/7371/computer-programming-services (last visited Oct. 8, 2017).

145. COMPUSTAT data is not uniformly available for all publicly listed companies. When data from the particular year that the license was reported was not available, I chose the closest year.

SEC filings. For private firms, this study relied upon ReferenceUSA and company websites to determine year of founding. It also excluded transactions with individuals from the analysis, as well as transactions involving firms for which I could not find founding year or revenue data, resulting in a match for about 45% of transactions.

IV. FINDINGS

What were the results from analyzing software patent transactions? This study finds that, despite recent legal developments that have reduced the enforceability of software patents, the market for software patents has remained remarkably robust, for reasons explored in depth below. Second, this Part finds that software patent transfers are supporting the transfer of technology—not just the transfer of liability or freedom from suit—but that both appear to be strong motivators for transactions.

A. THE MARKET FOR SOFTWARE PATENTS REMAINS ROBUST DESPITE A DECLINE IN THE ENFORCEABILITY OF SOFTWARE PATENTS

The first finding of this study pertains to the importance of the marketplace for diffusing software innovation between firms. Studying the market addresses several gaps in our understanding of software innovation. First, although most of the policy attention with respect to software patents has been focused on disputes about their quality, patterns of assertion, and infringement, the sales and licensing of software patents provide more direct insights into the transactional role software patents are playing, on a day to day basis, in stimulating and supporting innovation, or not.

Second, while much has been written about open modes of diffusing software innovation across firms borders, such as employment laws that prohibit the enforcement of non–compete agreements[146] or the open source software movement,[147] the paid market for software innovation as reflected in software patent licenses and sales represents a sizeable and important mechanism for technology transfer. Understanding the dynamic between open and proprietary innovation is an important step in ensuring adequate support for both models.

Finally, while there have been a number of significant policy developments in the realm of software patents in the past few years, their

146. *See, e.g.*, ORLY LOBEL, TALENT WANTS TO BE FREE: WHY WE SHOULD LEARN TO LOVE LEAKS, RAIDS, AND FREE RIDING (2013).

147. *See* Karim R. Lakhani & Eric von Hippel, *How Open Source Software Works: "Free" User-to-User Assistance*, 32 RES. POL'Y 923 (2003) (offering overview of the open source software movement).

impact on software innovation has not been clear. In general, software patents have become harder to enforce in recent years. The America Invents Act of 2011 introduced a host of new procedures to challenge the validity of issued patents.[148] These procedures have not been kind to software patents.[149] The Supreme Court's *Alice* decision in 2014 erected significant limits to patentable subject matter, making it harder to get patents over business methods and the abstract algorithms that are at the heart of software innovation.[150] Almost immediately, defendants began mounting "*Alice*" challenges to patents they were sued on, invalidating them in many cases.[151] The Supreme Court's decision in *Nautilus* decision has also made it easier to mount invalidity challenges to software patents that contain claims based on functional language, on the basis that they are "indefinite."[152] Holding all else equal, these developments would be expected to depress the market for software patents.

1. Patent Sales

Against this backdrop, the data tells a distinct story. The paid market for software innovation is robust: in a single year, the data show a software patent is equally or more likely to be sold (~2%) than it is to be litigated over its entire lifetime.[153] Rather than declining, the absolute number of software patent transfers has actually increased, from around 5,900 patents per year in 2012 to 8,900 patents per year in 2015, a 68% rise. (Figure 2). Although this analysis only extends through 2015, related work by Love carried out through 2016 reinforces the robustness of the market. Love finds

148. These include inter partes review (IPR), the covered business method transitional program (CBM), and post–grant review (PGR). *See* Joe Matal, *A Guide to the Legislative History of the America Invents Act: Part II of II*, 21 FED. CIR. B.J. 539 (2012) (explaining the rationale for and features of these procedures).

149. Brian J. Love & Shawn Ambwani, *Inter Partes Review: An Early Look at the Numbers*, 81 U. CHI. L. REV. DIALOGUE 93, 105–06 (2014) (finding petitions for inter partes review result in elimination of every challenged claim about twice as often as the same result for requests for inter partes reexamination).

150. Alice Corp. v. CLS Bank Int'l, 134 S. Ct. 2347 (2014). This case followed *Bilski*, which also raised the patentability bar for software inventions. The evolution of the subject matter requirements for software is traced in Kirk Teska, *(The Unfortunate) Future of Software Patents Under 35 USC § 101 and § 112*, 16 J. HIGH TECH. L. 394 (2016).

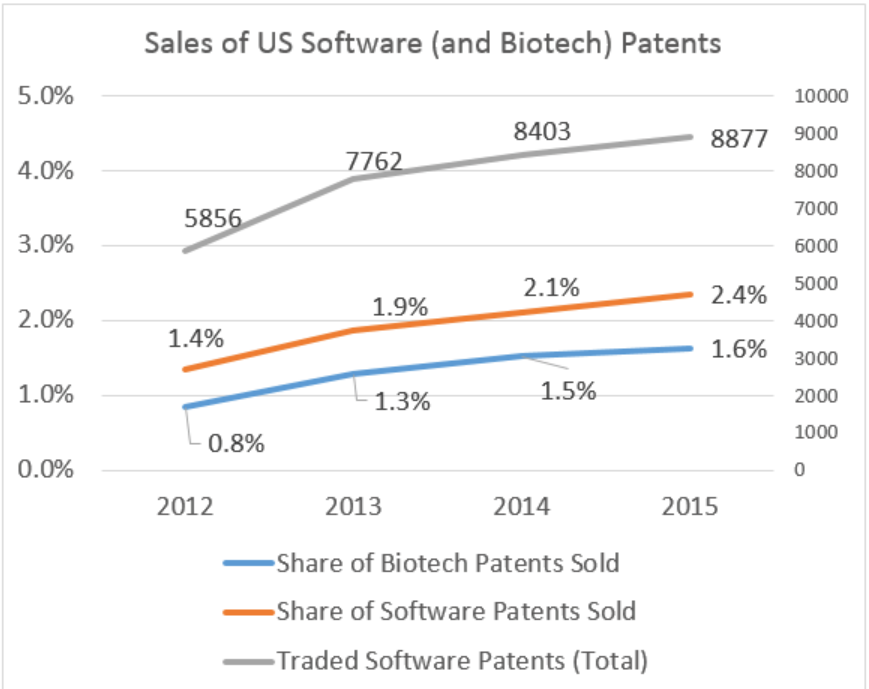
151. Jasper L. Tran, *Software Patents: A One-Year Review of* Alice v. CLS Bank, 97 J. PAT. & TRADEMARK OFF. SOC'Y 532, 540–41 (2015) (showing the district courts, PTAB, and Federal Circuit invalidated 82.9% of patent applications in the year following *Alice*).

152. However, it is unclear what impact the decision has actually had on court cases. *See* Jason Rantanen, Teva*,* Nautilus*, and Change Without Change*, 18 STAN. TECH. L. REV. 375, 377–80 (2015) (describing the *Nautilus* case and its lack of impact).

153. About 1–2% of all patents are ever litigated. *See* Lerner et al., *supra* note 37.

that, from 2012 to 2016, the number of packages of software patents sold on the brokered market was highest in 2016, though the median asking price per asset declined over this period close to 20%.[154]

Figure 2: The Sale of US Software (and Biotechnology) Patents (2012–2015)[155]



To contextualize these findings and explore the possibility that this increase reflects changes in the number of patents—or other changes outside the patent system—this study considered not only the absolute number of patents being transferred, but the relative rate of software patent transfers as compared to the total number of in–force patents. This study also compared software patent and biotechnology patent transfer rates.[156] These calculations reinforce the robustness of the software patent market—reflecting a rise in the transfer rate from 1.4% in 2012 to 2.4% in 2015, and far outstripping the rate of biotechnology patent transfers, which totaled

154. Love et al, *supra* note 40, at 33 & tbl.13.

155. This figure is calculated based on unique patents. I did not control for continuations, which may be more common among biotechnology patents than software patents.

156. I chose the biotechnology industry as a point of comparison because the biotechnology sector is often held up as an example of a well–functioning innovation market, in which larger firms are well–poised to commercialize and absorb smaller firms (or their technology) and bring it to market.

0.8% to 1.6%. When comparing the top transactions in both sectors, the data showed that the size of the average portfolio of transferred software patents was much larger than that of transferred biotechnology patents.

This finding is significant for at least two reasons—first, it reinforces that software patents are actually much more likely to be transferred than litigated. Scholars and policymakers, in contrast, have concentrated far more on the litigation of software patents than their transfer. The scholarly community should turn more attention to this set of patent transactions, and the dynamics between sales and litigation. Second, the data show that the market for software patents remains robust, and has even grown, in the face of significant legal developments calling into question the enforceability of software patents. What is behind the demand for software patents? This Article discusses three possible explanations.

a) Bargain Shopping for Software Patents

Although detailed transactional data is hard to come by, one reason for the uptick in patent transactions may be that the price per patent has gone down. According to one estimate,[157] from 2014 to 2015, asking prices were down about ~$90,000 per patent, from $280,000 per asset to $190,000 per asset, even as sales increased.[158] The increased sales volume may reflect opportunistic buying on the part of those who want to decrease the risk of patent assertions and perceive a buying opportunity. In 2016, the patent buying consortia IP3, representing IBM, Apple, Google, Microsoft and a number of the other top targets of patent litigation announced that it would be soliciting offers to sell patents to the consortia.[159] Building on an experiment to buy patents directly from patent holders carried out by Google the previous year[160] and the efforts of defensive aggregating intermediaries,[161] the group is exercising monopsony power to "buy in bulk." This shift in purchasing strategy further reduces the group members' own costs and cuts out the middlemen of patent litigators, patent brokers, and patent assertion entities (PAEs). As the enforcement climate grows less favorable to patent holders, the option of monetizing through direct sales

157. Richardson Oliver Law Grp., Presentation to the IPBC in Barcelona (June 6, 2016) (on file with the author).

158. *Id.* (reporting a 23% increase in sales of all patents from 2014 to 2015, larger than the increase that this study observed among software patents during that period of time).

159. Richard Lloyd, *The Timing is Perfect for IP3's Patent Buyers; For Sellers the Picture is Far Less Rosy*, IAM MEDIA (May 20, 2016), http://www.iam-media.com/Blog/Detail.aspx?g=3cbec828-2e85-4746-b422-772e5f294aa4.

160. *Id.*

161. RPX and Allied Security Trust (AST) are such examples.

rather than assertion may be attractive to both parties, even at lowered prices. In addition, given fixed budgets for the purchase of patents, when the cost per patent declines, the volume of patents sold goes up.

b) Defensive Rather Than Offensive Acquisition of Portfolios of Software Patents

Another driver of software patent transactions is the purchase of patents for defensive or strategic—rather than offensive—purposes.[162] Patents create freedom to operate in at least three ways. First, the presence of an arsenal of patents, and closely related technology, deters attacks by competitors because it enables the owner of the arsenal to bring a countersuit if threatened. Second, patents provide trading assets that allow companies to exchange technology through cross–licensing. In both contexts, the quantity of patents held in a portfolio is just as, if not more, important as the quality or enforceability of any individual patent. Thus, while a single patent or group of patents might now appear to be invalid under the *Alice* decision, it is likely that within an entire portfolio, there are still enforceable assets, and the costs of determining the difference on a patent–by–patent basis is often prohibitive. Likewise, in a license negotiation between two parties, even though one patent may be a strong candidate for invalidation under an AIA procedure, challenging an entire patent portfolio—which may number in the hundreds—is impractical. Thus the decline in enforceability of individual patents has not necessarily translated into a greater freedom to operate, meaning there is still a strong need for additional patent assets.

A third defensive driver of patent transactions is buying or licensing patents in order to take them off the market to avoid being sued over their infringement later. "Defensive buying" consortia such as Rational Patent Exchange (RPX), the Open Invention Network (OIN), and License on Transfer (LOT) Network purchase or secure the rights to purchase or license patent assets that they believe pose risks for their members.[163] The patents at stake are often sourced from operating companies, reflecting the importance of operating companies as sources of patents on the market.[164]

162. *See, e.g.*, James M. Rice, *The Defensive Patent Playbook*, 30 BERKELEY TECH. L.J. 725, 726 (2015).

163. Chien, *supra* note 7 (describing these models in detail). As of September 2017, the License on Transfer network included over three quarters of a million assets. *See Eliminate the Patent Troll Threat*, LOTNET.COM, http://lotnet.com/ (last visited Oct. 8, 2017) ("LOT members are immunized against 785,462 worldwide assets in the network").

164. Love et al., *supra* note 40, at 22; *see also* Chien, *supra* note 48, at 313–14 (exploring why operating companies sell their patents and tracing the assertion by PAEs of patents once originally owned by operating companies).

Demonstrating the spirit of the maxim "an ounce of prevention is worth a pound of cure," the rights to assets are bought for a fraction of what they would cost to defend against in a lawsuit.

c) Software Eats the World

Finally, the value of a patent is a product not only of its legal validity, but the economic value of the technology it covers. A patent that conforms to all the legal requirements of patentability but covers a worthless technology has little value. Similarly, a portfolio of patents over a valuable technology, even if the validity of some of the patents is contestable, can be worth millions. While the legal enforceability of software patents has declined recently, there does not appear to be any corresponding decline in software innovation.[165] Growth in the U.S. software sector has outpaced overall economic growth over the past few decades.[166] Google and software company SAS are among the best places to work in America,[167] and the stocks of software and internet companies like Netflix, Electronic Arts, Activision, and Amazon lead the stock market.[168] The market for software patents reflects the vibrancy of the software industry to a greater degree than it does the legal enforceability of software patents. In this sense, software innovation could be said to be happening not because of, but in spite of or unrelated to software patents.

2. Additional Evidence from Licenses

The importance of the market for software–based innovation can be gauged not only through sales of software patents but also through agreements for software innovation. As described earlier, this study considers agreements reported to the SEC by public companies that deem the agreements to be "material" events that could impact the company's

165. And in fact, software innovation is increasingly leading even in traditional, manufacturing sectors of the economy. *See* Branstetter et al., *supra* note 3.

166. Robert J. Shapiro, *The U.S. Software Industry as an Engine for Economic Growth and Employment* 1, 7 (Georgetown McDonough Sch. of Bus., Research Paper No. 2541673, 2014), http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2541673 (finding that, from 1997 to 2012, growth of the software industry outpaced growth in the rest of the economy, capturing an increasing share of national GDP, and contributing 3.2% of GDP in 2012).

167. *See, e.g.*, *100 Best Companies to Work for*, FORTUNE, http://fortune.com/best-companies/ (last visited Oct. 8, 2017) (listing Google as the number one top place to work from 2014 to 2016 and SAS Institute among the top ten in that period).

168. Laurie Kulikowski, *The 10 Best S&P 500 Stocks in All of 2015*, THESTREET (Oct. 3, 2015, 11:31 AM), https://www.thestreet.com/story/13306053/1/the-10-best-s-p-500-stocks-in-all-of-2015.html.

stock price. As such, it is important to note the limited nature of this sample, as it excludes many agreements to license software innovation.

Keeping this caveat in mind, the SEC data supports the importance of software in technology transactions among a variety of different industries. According to ktMINE's version of the SEC database, about 23% of all technology agreements[169] reported to the SEC between 2000 and 2015 (1,431 out of 6,109) involved the transfer of software.[170] That is to say, nearly a quarter of important technology agreements to public companies were software agreements. To put that number in context, software companies contributed about 3% to GDP in 2012.[171] That the share of software technology transactions is greater than software's contribution to GDP is unsurprising, but the extent of this difference is dramatic.

How were software agreements distributed across and within industries? Innovation scholars have long discussed the contrast between "cumulative" innovation areas like software in which many, even thousands, of incremental innovations may be embodied in a single product, and "discrete" biopharma innovations, which may be covered by just a handful of patents.[172] The differences in these two types of innovation have strained our unitary patent system, which does not permit discrimination based on technology.[173] However, to the extent that cumulative, software–based innovation is widespread across sectors, these distinctions may be blurring.

From 2000 to 2015, this study finds material software agreements were spread among a variety of different technology areas, with the largest numbers of agreements covering business services, internet, telecommunications, and health care technologies. (See Appendix, Figure A1) The broad distribution of software agreements further demonstrates that software innovation is not restricted to certain sectors, but is shaping our economy more generally.[174]

169. As described above, these agreements include joint development, cross–license, manufacturing/intangibles agreements, software agreements and other agreements, and exclude franchise, distribution, service, marketing, and asset purchase agreements.

170. This is based on ktMINE's designation of the agreement as a "software" agreement.

171. Shapiro, *supra* note 166, at 17–18 (finding that, from 1997 to 2012, growth of the software industry outpaced growth in the rest of the economy, capturing an increasing share of national GDP, and contributing 3.2% of GDP in 2012).

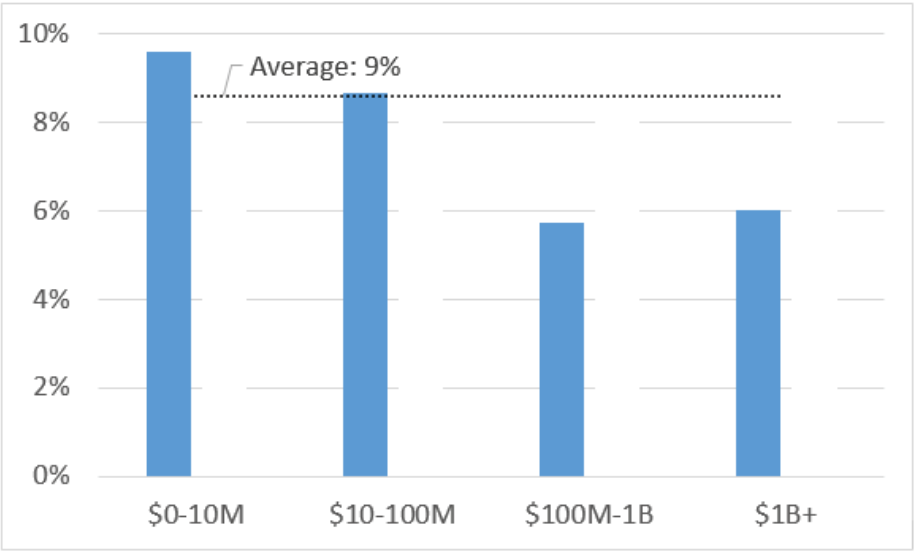
172. *See, e.g.*, Cohen et al., *supra* note 43.

173. Stefania Fusc, *TRIPS Non-Discrimination Principle: Are* Alice *and* Bilski *Really the End of NPEs?*, 24 TEX. INTELL. PROP. L.J. 131 (2015).

174. *See* Branstetter et al., *supra* note 3, at 20.

What about the distribution of agreements within industries? The data discussed thus far, about the number of technology agreements, and the share of them that are software agreements, do not measure the likelihood that any individual company is to enter into a material agreement covering software. To measure this, this study looked specifically at "pure" software companies and the extent to which they did or did not report material software agreements. SEC filings showed that a modest share of all public companies,[175] around 9%, had reported one or more software agreements. (Figure 3). The smaller a company was, the more likely it was to have reported an agreement.

Figure 3: Share of Pure Software Companies (by Annual Revenue Band) Reporting a Material Technology Agreement to the SEC (2000–2015)



While the findings described above provide some basic facts about the likelihood, prevalence, and distribution of paid transfers of software innovation, they do not address the substance of these transfers. When a software patent is transferred from one firm to another, what is sold, exactly? When a company signs an agreement to share software innovation with another, what exactly is it sharing, and on what terms? These questions are important to address as not all transfers of software innovation are created equal, nor do they confer the same social costs and benefits. In the following paragraphs, this Article consider patterns of patent sales, as well as SEC reported patent licenses, addressing where possible the extent to

175. This information was tracked by COMPUSTAT.

which the transfer or license represents a transfer of technology or a transfer of liability.

B. SOFTWARE PATENT SALES SUPPORT BOTH TECHNOLOGY AND LIABILITY TRANSFERS

When Google bought Motorola and its patents in 2011, it was primarily for its ability to protect the Android ecosystem,[176] but the transaction was unusual—typically when a company buys another, it is in order to buy the business, including the technology and innovation that may be protected by patent. But the wide variety of ways in which patents be used, including for protection (freedom to operate), signaling, trading, or protecting the underlying technology through exclusion[177] gives rise to a wide variety of motivations for patent sale. One way to discern the purpose of sale is to look at its terms and downstream uses. The pattern of a transfer may also reveal the motives of the buyer, in particular with respect to the relative ages of the parties. For example, patents can support the sale of the technology of a young company to an older company better positioned to commercialize the technology, helped by intermediaries.[178] Conversely, patents may be transferred from an older to a younger company when the younger company is infringing the patent and seeks freedom from suit, or a unit of the older company is divested to a younger company.

Although the terms of patent sales are generally not publicized, information about large transactions is often available. Figure 4 lists the top ten sales of software patents (by number of patents) recorded from 2012 through 2015. Reviewing public disclosures about each "top transaction," about half appear to have been associated with defensive or otherwise liability–shifting motivations, while the remainder supported the broader transfer of a technology business. Strikingly, in all of the transactions, assets moved from an older to a younger company. After identifying this pattern, this study probed whether it held among transfers for which information was available. It did, with software patents between two and three times more likely to be transferred from an older to a younger company than vice versa. This finding contrasts sharply with the commercialization story of patents in which a young upstart sells its patents to an established incumbent, as further discussed below. The results were robust across every

176. *Facts About Google's Acquisition of Motorola*, GOOGLE, https://www.google.com/press/motorola/ (last visited Oct. 8, 2017).

177. *See, e.g.*, Stephen Yelderman, *Coordination-Focused Patent Policy*, 96 B.U. L. REV. 1565 (2016) (offering an overview of these uses).

178. *See, e.g.*, James F. McDonough III, *The Myth of the Patent Troll: An Alternative View of the Function of Patent Dealers in an Idea Economy*, 56 EMORY L.J. 189, 190 (2006); Feldman & Lemley, *supra* note 31, at 138.

year studied and both individual patent transfers and transactions. This study also found that the skew in favor of "old to young" transfers was much more pronounced among software patent transfers than biotechnology patent transfers. Below, this Article delves more deeply into the top ten transactions and explore whether patterns observed in this small dataset are generalizable more broadly.

Figure 4: Top 10 Software Patent Transfers (2012–2015) and Years of Founding of Transferors and Transferees

Transaction	Software Patents Transferred[179]	Year of founding of Transferor	Year of founding of Transferee
IBM to Globalfoundries Inc.	2240	1911	2009
HP Inc. to TCL Corporation	1123	1939	1981
Lenovo Group to Alphabet Inc.	834	1984	1998
Fujitsu and Panasonic to Socionext	820	Fujitsu: 1935; Panasonic: 1918	2015
IBM to Lenovo Group	783	1911	1984
HP to Qualcomm	599	1934	1985
IBM to LinkedIn	516	1911	2002
IBM to Twitter	495	1911	2006
IBM to Facebook	414	1911	2004
Eastman Kodak to Intellectual Ventures Management	310	1888	2000

1. Sales That Transfer Liability

One of the most striking things about the list of top ten software patent transfers is that five involve the transfer of patents from IBM to other companies. For years, IBM has been the top recipient of US patents, so its dominance of the top seller list is not necessarily surprising. Three of the five transactions of IBM patents, to the young technology companies of

179. It is worth noting that these counts reflect only the transfer of software patents, and the actual transactions may have also encompassed non–software patents.

LinkedIn, Twitter, and Facebook appeared to fit the profile of "liability" rather than "technology" transfers. In 2013, IBM reportedly sent a letter to Twitter claiming that it was infringing several of IBM's patents and invited the company to "sort it out or face the consequences."[180] Practicing a well–known tactic,[181] IBM approached Twitter during one of its most vulnerable times, when it was trying to go public.[182] Ultimately, Twitter bought many more patents, perhaps as many as nine hundred, than the handful that it was alleged to be infringing.[183] This suggests that Twitter thought it would be useful to have not only freedom from the patents specifically asserted against it, but also assets that it could use to ward off other threats. According to reports, prospective litigation also led Facebook to acquire at least 400 patents from IBM.[184] LinkedIn's purchase of IBM patents also appears to have been motivated by a desire to avoid legal liability, which could have been asserted by IBM or a buyer of its patents.[185]

Several others of the top ten purchases appear to have had defensive intents. For example, Intellectual Ventures (IV) purchased a large number of patents from defunct photography company Eastman Kodak. According to public reports, the deal was organized by IV and RPX Corporation on behalf of twelve intellectual property licensees, with each licensee receiving rights with respect to Kodak's digital imaging patent portfolio and related patents.[186] In another apparently defensive move, when Alphabet (Google) sold Motorola's mobile business to Lenovo, it retained the patent assets,

180. Brid-Aine Parnell, *Twitter Avoids IP Face-off with Big Blue, Will Buy 900 IBM Patents*, REGISTER (Feb. 3, 2014), http://www.theregister.co.uk/2014/02/03/twitter_ibm_patents/.

181. *See* Robin Feldman & Evan Frondorf, *Patent Demands and Initial Public Offerings*, 19 STAN. TECH. L. REV. 52, 73–79 (2015) (finding the percentage of companies surveyed with patent claims filed against them jumped from 10% before S-1 filing to 40% shortly before or after the IPO).

182. Parnell, *supra* note 180.

183. *Id.*

184. Gene Quinn & Steve Brachmann, *Facebook and Twitter: Patent Strategies for Social Media*, IPWATCHDOG (Feb. 14, 2014), http://www.ipwatchdog.com/2014/02/14/facebook-and-twitter-patent-strategies-for-social-media/id=48004/.

185. *See Patent Market Tracker Fall 2015 Key Trends*, INNOGRAPHY (2016), https://www.innography.com/public/upload/files/general-files/Innography-Patent-Market-Tracker.pdf.

186. Andrew Martin, *Kodak to Sell Digital Imaging Patents for $525 Million*, N.Y. Times (Dec. 19, 2012), http://www.nytimes.com/2012/12/20/business/kodak-to-sell-patents-for-525-million.html.

which were assigned back to Alphabet when Google was reorganized.[187] (Figure 5, Lenovo Group to Alphabet Inc.).

As discussed earlier, scholars have previously considered the impact of the patent sales on the propensity of patents to be litigated. While my research on the topic did not find an increase in the likelihood of litigation upon transfer,[188] Galasso and his colleagues found that it depended on the context. Transfers from individual inventors to larger entities had a reduced likelihood of litigation, on average, while transfers from larger to certain smaller entities were correlated with an increased likelihood of litigation.[189] But while the transactions just described appear to be motivated by the desire to avoid patent enforcement, one transaction in the top ten appears to have transferred liability in another direction, to a party with advantages in enforcement and licensing. In 2014, Qualcomm purchased hundreds of HP patents covering the company's mobile computing technology.[190] Few financial details or intentions with respect to the patents involved in the deal were released,[191] but Qualcomm makes about a third of its revenue from licensing patents,[192] and it is plausible that the assets were being purchased to support this type of revenue generation.

2. Sales That Transfer Technology

While the transfers just described supported liability transfers, in both directions, other top ten transfers supported transfers of entire businesses and technologies. For example, chip manufacturing has long been among IBM's many activities, but has caused IBM to lose money in recent years.[193] In 2014, IBM entered into a deal to transfer its facilities to GlobalFoundries, which would continue to operate and produce chips for IBM in exchange

187. Claire Miller & David Gelles, *After Big Bet, Google Is to Sell Motorola Unit*, N.Y. TIMES: DEALBOOK (Jan. 29, 2014, 4:13 PM), https://dealbook.nytimes.com/2014/01/29/google-seen-selling-it-mobility-unit-to-lenovo-for-about-3-billion/.

188. *See* Chien, *supra* note 27, at 320.

189. Galasso et al., *supra* note 52.

190. Jeffrey Burt, *Qualcomm Buys Palm Patents from HP*, EWEEK (Jan. 24, 2014), http://www.eweek.com/mobile/qualcomm-buys-palm-patents-from-hp.html.

191. *Id.*

192. *See* Qualcomm, Annual Report (Form 10-K) (Sept. 27, 2015), http://investor.qualcomm.com/secfiling.cfm?filingID=1234452-15-271&CIK=804328 (showing that about eight billion out of the firm's twenty–five billion in revenue is from licensing).

193. Joel Hruska, *IBM Sells Chip Business to GlobalFoundries for $1.5 Billion (Updated)*, EXTREMETECH (Oct. 20, 2014), http://www.extremetech.com/computing/192430-ibm-dumps-chip-unit-pays-globalfoundries-1-5-billion-to-take-the-business-off-its-hands.

for around $1.5 billion in cash.[194] As part of the deal, over 2,000 patents were transferred to GlobalFoundries. (Figure 5). In another divestiture, IBM sold its personal computer business, including a large number of IBM's patents, to Lenovo group[195] for $1.75 billion.[196] Other patent transactions in the top ten fit the pattern of being part of a larger business transfer, such as HP's Palm unit to TCL,[197] and the combination of assets of Fujitsu and Panasonic to form Socionext, a chipmaker.[198]

3. Patterns of Transfer—From Old to Young and Rich to Poor

Although each transfer in the top ten had its own motivation, strikingly, they all follow a similar pattern. In every case, the software patents were being transferred from an older company to younger company. (Figure 5). More often than not, the transfer also reflected movements from the company with greater revenue to the company with less revenue. Because the top transactions of any set are often unique, and cannot be generalized to the entire set, this study took additional steps to investigate whether the transfer patterns observed at the top—from older to younger companies, and from companies with more revenue to companies with less revenue—were observed among transactions in general. Using the methods described above, this study was able to match 45% of transfers. Because this study had to exclude transactions to and from individuals from the analysis, as well as companies that did not have an English–language website from which founding year data could be easily determined, the analyzed transactions are likely skewed toward larger, more successful companies. For the revenue data, the match rate was also about 44%, because all private companies were excluded from the analysis due to the lack of reliable sources of private company revenue. The findings are presented in Figures 6 and 7.

194. *Id.*

195. *The IBM/Lenovo Deal: Victory for China?*, KNOWLEDGE@WHARTON (Jan. 14, 2005), http://knowledge.wharton.upenn.edu/article/the-ibmlenovo-deal-victory-for-china.

196. *Id.*

197. Eric M. Zeman, *TCL to Revive Palm with Help from the Tech Community*, PHONE SCOOP (Jan. 6, 2015), http://www.phonescoop.com/articles/article.php?a=15128.

198. *Id.*

Figure 5: Transfers of Software Patents by Age of the Parties (2012–2015 Transactions; N = 13,904)[199]

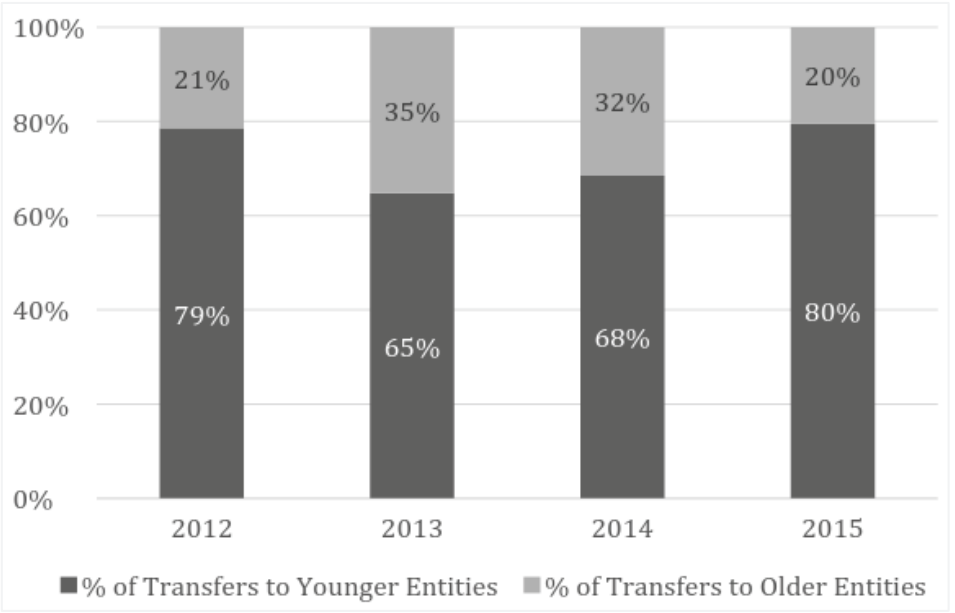
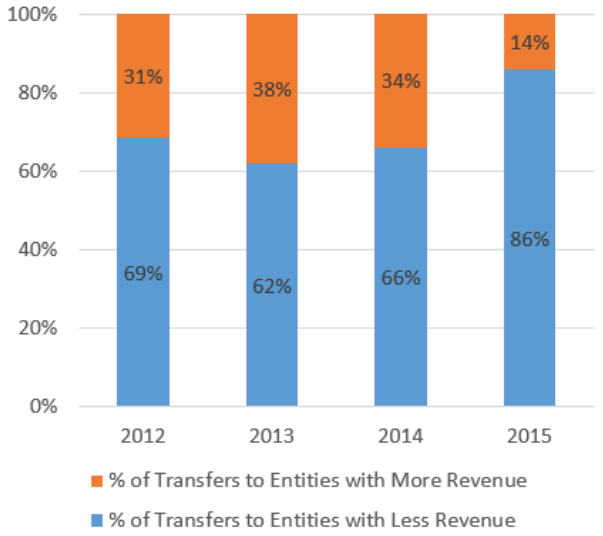


Figure 6: Transfers of Software Patents between Public Companies by Revenue of the Parties



The results are striking. The patterns of old to young as well as all higher to lower revenue company software patent transfers were observed not just

199. The data represents 45% of recorded software patent transfers.

among the top sales, but more generally as well. Across the dataset, sales of software patents were between two and three times more likely to be from an older company to a younger company (73%) than from a younger company to an older company (27%). The difference between the observed distribution and a distribution in which transfers were equally likely to go from a younger to an older entity and from an older to a younger entity was statistically significant in every single year of the sample.[200] To rule out the possibility that the results were unduly skewed by transactions involving large numbers of patents, I also ran statistical tests at the deal level, rather than the individual patent level. The results were similar.[201] Among transactions between public companies of different revenue levels, the majority of patents also moved from higher revenue to lower revenue companies. Sales were, on average, more than four times more likely to be from a company with more revenue to a company with less revenue (71%) than vice versa (29%). This difference was maintained across the years of the study, and was statistically significant in each year at both the individual patent transfer level and the deal level.

To test how unique these patterns were, and whether they were true of patent transfers in general rather than mere artifacts of software patent transactions, the study replicated the analysis among a subset of biotechnology patent transfers.[202] Biotechnology patents were also more likely to be transferred from older, higher–revenue companies to younger companies with lower revenues. But the transactions were more evenly split among transfers to older and younger companies, and those with higher and lower revenues. 47% of biotechnology patent transfers were to older companies, and 53% to younger companies. 45% of biotechnology transfers were to public companies with more revenue, and 55% to companies with less revenue. Neither of the differences between the observed values and an equal distribution were consistently statistically significant across the tested

200. This study used a standard chi–square test to examine the null hypothesis that, in a given year, software patent transfers were equally likely to go from an older to a younger company as vice versa, yielding p–values of 0 to 1.6197E-81. A p–value of less than .05 is generally interpreted as an indication that the null hypothesis can be rejected (making it statistically significant), while a value greater than 0.10 is viewed as showing that any differences are not statistically significant. For the exact p–values, see the Appendix.

201. On average, 60% of deals were from an older to a younger company, and 40% were from a younger to an older company. Running a chitest (using Excel's CHITEST function) that compared the observed distribution to an even distribution, the p–values were 0 to 8.17389E-54.

202. There was N=1093 biotechnology patent transfers for the revenue analysis, and N=995 biotechnology patent transfers for the age analysis.

years,[203] in contrast to the statistical significance of differences among software transfers. This may reflect, in part, the relatively fewer observed biotechnology transfers.[204]

While striking at first blush, the movement of software patents from older, relatively higher revenue companies to younger, lower revenue companies has several possible explanations. For several decades there has been a "patent arms race" among technology companies, as companies have filed patents early and often to deter suits by competitors or other operating companies.[205] But as a company matures and evolves, its needs change, including its need to keep all of the patents in the portfolio. Rather than just retiring the patents, companies can sell them to others who can make better use of them. Younger companies with rapidly increasing revenues, in turn, need patents to protect against potential patent demands: indeed, companies like Twitter, Facebook, and LinkedIn have found the option to buy patents attractive. These types of transfers benefit both parties, as patent holders are able to recoup some of the costs of R&D and fund additional innovation, and patent–receiving companies can avoid delays and uncertainty at the USPTO and buy—rather than build—their own patent portfolios.

When a patent transfer is part of a larger business transfer the acquired business is rewarded not only for its existing revenue, but for its investment in future products and services. It provides more flexibility for the transferor to develop the technology, either on its own or with commercialization partners. Because they are portable, portfolios of patents can provide scaffolding and support for business transactions, making it easier to transfer technology and the rights to exclude others from practicing them.

But the profile of rent transfers from small to large companies, without any accompanying technology, also supports criticisms that software patents are effectively a tax on innovation. Though younger companies get patents, they must pay for them, forcing a transfer of wealth from the relatively younger to the relatively older company. When only patents, not technology, are transferred, the welfare effects can be ambiguous, as the

203. *See infra* Appendix.

204. Biotechnology patent transfers differed in other ways from software patent transfers. Among the top ten, almost all involved less than 100 biotechnology patents, while among top transfers of software patents, most involved more than 500 software patents. This skew in size of top transactions is reflected in a much larger average transaction size of 7.5 software patents versus 2.4 biotech patents per transfer, although as described below, for both types of patents the median and mode number of patents per transaction was 1.0.

205. *See* Chien, *supra* note 48 (offering an overview of the industry and firm–level dynamics shaping the marketplace for high–tech patents).

gain to the larger patent holder must be weighed against the cost to the smaller patent implementer without the exchange of technology. When the patents are transferred and then asserted against independent development and practice of the patent, the "tax" can be widespread, encompassing not only the independent developers but also the users of technology.[206]

If patent sales have been in support of both technology and liability transfers, what about patent licenses? The next section describes the analysis performed to probe the motivations for licenses, and the results found by the study.

C. SOME SOFTWARE PATENT LICENSES ARE FACILITATING THE TRANSFER OF TECHNOLOGY

While software patent sales can provide some insight into the extent to which technology and rights are distributed, parties are not required to disclose, much less register, how they intend to use the transferred patent. A more granular perspective on the substance of the innovation transfers can be gleaned by looking at licenses in which licensor and licensee usually spell out their intentions for the patents. The problem with licenses, however, is that they are largely not available for inspection. In the following analysis, I skirt this obstacle by relying on material technology licenses recorded with the SEC, though it bears repeating that these licenses are highly selected and unrepresentative of licenses in general. The remaining paragraphs describe the results of the in–depth review of these agreements for indicia of the software innovation being transferred through them.

Among material patent agreements recorded with the SEC, patents are supporting the transfer of technology, not just freedom from suit. Among licenses where patents are "core," patents generally support the transfer of trade secrets, know–how, or other proprietary information, consistent with theories of how patents resolve the Arrow information paradox. However, non–patent proprietary assets—in particular code and trade secrets—are more commonly transferred than patents. In addition, the presence of intellectual property in the agreement does not necessarily impact the exclusivity profile of the license—that is to say, licenses were just as likely to be exclusive or non–exclusive regardless of intellectual property protections. This suggests that in many cases contract law, rather than patent or other intellectual property, may be doing the heavy lifting.

206. *See, e.g.*, Chien & Reines, *supra* note 27.

1. SEC Software Patent Licenses

Though studies described earlier have documented the use of licenses to support the transfer of both technology and liability, current research suggests that in recent years, when licensees are approached to take a license, they walk away from the deal with little more than a way to avoid costly litigation.[207] Recent studies of patent licensing cast patent licenses in a similar light, characterizing them as always conducted in the shadow of litigation rather than, for example, the shadow of competition.[208] To test the extent to which patent licenses were merely providing a shield from litigation, with little additional benefit, I considered the terms of licenses. I found some evidence consistent with the idea that patent–related clauses within agreements primarily served the role of confirming or shifting liability: in the majority of software tech licenses, patents were mentioned not as the subject matter of the transfer but as part of an indemnity or limitation of liability clause.[209]

---

207. Feldman & Lemley, *supra* note 31, at 137 ("[F]ind[ing] that very few patent license demands actually lead to new innovation; most demands simply involve payment for the freedom to keep doing what the licensee was already doing.").

208. Jonathan Masur, *The Use and Misuse of Patent Licenses*, 110 NW. U. L. REV. 1115 (2015); William F. Lee & A. Douglas Melamed, *Breaking the Vicious Cycle of Patent Damages*, 101 CORNELL L. REV. 385 (2016).

209. *See, e.g.*, the following mentions of patents within agreements:

> 5 INDEMNIFICATION
>
> 5.1 Agilent shall defend and indemnify Ansoft and hold it harmless from any and all losses, damages, costs and out-of-pocket expenses, including reasonable attorneys' fees, incurred by Ansoft that result from any claim, lawsuit, proceeding, or other action, whether legal or equitable, by a third party alleging that the unmodified Agilent HFSS Software Products or the DomainName infringes any copyright, trade secret, patent, or other intellectual property right, anywhere in the world. Counsel provided by Agilent to represent Ansoft shall be mutually acceptable to both parties. Ansoft may participate in any such claim at its own expense.

Exhibit 10.12, Agilent HFSS Technology License and Transition Agreement (May, 1 2001), https://www.sec.gov/Archives/edgar/data/849433/000095015203007071/j0226301exv10w12.txt

> 10.10 No Other Licenses. Nothing in this Agreement will be deemed to grant, by implication, estoppel, or otherwise, a license under any of Parthus's existing or future patents; however, Parthus agrees that it will not assert any of its rights under such patents against Licensee or its Customers based on the manufacture, use, sub-license or distribution of

However, in cases where patents were considered "core," they were not licensed alone—the patent rights were accompanied by transfer of know–how, code, and other proprietary assets. The vast majority (98%, 240/245) of these patent licenses included trade secrets of some form, or some sort of computer code (generally object code), source code, library, bug fix, and/or executable (95%, 232/245). That is to say, in contrast to some evidence that patent licenses almost never include other forms of technology transfer, this study found the opposite—that the patent licenses in the study almost always included trade secrets or source code, and often both. (Figure 7).

Figure 7: Transfers Among Software–Patent Agreements (N=245)

The transfer of technology, as opposed to naked patent rights, was striking. In contrast with *licensor*–initiated licenses, the significant technology agreements this study studied largely reflected mutual rather than one–sided interest, and the ex ante rather than ex post licensing of technology. This suggests that patent licenses play an integral role with respect to *both* types of transfers.

This study also tested the theoretical roles of patents by examining actual agreements. Consistent with prospect theory, within the agreements patents provided a way to identify the subject matter of the transfer. In the following example clause from a license, patents were used to designate not only the technology being transferred, but also the technology *not* being transferred:

> the Licensed Products as permitted by this Agreement. Nothing contained in this Agreement shall be construed as conferring by implication, estoppel or otherwise upon either party hereunder any licenses or other right except the licenses and rights expressly granted hereunder to a party hereto.

Exhibit 10.21, Parthus Technologies PLC License Agreement (Sept. 30, 2002), https://www.sec.gov/Archives/edgar/data/1173489/000095016802002982/dex1021.htm.

> (i) TECHNOLOGY – Technology, as used herein, shall mean and refer to the algorithms, software and hardware designs, and methods relating to the field of image processing, specifically to the efficient coding and compression, decoding and decompression of video images, described in Differential Order Video Encoding System, US Patent #5,739,861, issued Apr.14, 1998. Japan Patent #3441736 issued Sept. 2, 2003. Canada Patent #2,252,545, issued July 13, 2004 and Patents Pending in E.U. and Korea, as well as certain related trade secrets, including invention, know-how, trade secret, function, design and any other features related to software that embody or are based upon the patents referred to herein and/or other proprietary intellectual property contained in Source Code. The term "Technology" shall not include, mean or refer to, and nothing contained anywhere in this Agreement shall confer or be deemed to confer upon ICOP any rights in or to, any of the algorithms, software and/or hardware designs, and methods relating to the field of image processing described in US Patents 5,164,819 (Method and System for Coding and Compressing Color Video Signals) issued November 17, 1992, and US Patent 5,448,296 (Variable Parameter Block Coding and Data Compression System) issued September. 5, 1995.[210]

It is difficult to know in the abstract whether a given agreement would have been signed without a patent. Besides showing up in an agreement, before the point of the transaction, a patent may have motivated the initial invention and supported the inventions' subsequent disclosure. What about in the example above? One might argue that the deal would have been much harder to reach in the absence of the patents, given the disclosing party's strict delineation of rights. In addition, the patent's terms defined the scope of the agreement, making it easier for the parties to transact. In some of the agreements, the definitional role of patents extended not only to the subject matter of the technology, but also to other terms of the agreement, such as its duration.[211]

However, patents may cut the other way too. The presence of a patent can lead to deals *not* getting done, insofar as it widens the gulf between the

210. *Software Decode License Agreement between Showlei Associates and ICOP Digital*, ONECLE INC. (Jan. 7, 2005), http://contracts.onecle.com/icop/showlei.lic.2005.01.07.shtml.

211. *See, e.g.*, Digital Audio System License Agreement (Professional Encoders) between Dolby Laboratories and Scopus Network (Aug. 2003) ("Section 6.01 - Expiration of Agreement: Unless this Agreement already has been terminated in accordance with the provisions of Section 6.02, this Agreement shall terminate five years from effective date or with the expiration of the last patent, whichever is first, and thereafter is renewable at LICENSEE's request at terms and conditions in force at the time of renewal").

patent holder, who may view the technology as that much more valuable because of the patent, and the prospective licensee, who cares only about the technology. When surveyed about why deals do not get done, licensing executives have pointed to the inability to reach agreement on price as the top reason.[212] Transactions involving IP assets are perceived as being more complex and costly to evaluate.[213]

In addition, in some subset of cases, parties who are determined to transact will figure out ways to do so, with or without patents. After all, in the majority of SEC software agreements, patents were not core. The next section provides additional context for understanding the role of patents, and intellectual property in general by comparing other types of transfers, and the impact of the presence of IP on exclusivity provisions.

2. *Software Patent Licenses Are Frequently Exclusive and Include Non–Patent IP Protections*

If patent rights were *not* being transferred in the majority of software agreements, what *was* being transferred? This study relied on codings by ktMINE to probe this question. Although patents were core to the transfer in about 34% of software agreements (480/1,419), other forms of intellectual property and proprietary technology were *more* prevalent and likely to be transferred. Trade secrets, proprietary rights, know–how, or related rights were core to 38% of the agreements,[214] while various forms of software—executables, source code, programs, bug fixes, libraries, operating systems, algorithms, and other software building blocks—were transferred in 88% of cases.[215] Copyright provisions were also pervasive, specifically showing up in about 31% of agreements, a number that potentially understates the importance of copyright given its automatic nature. A combination of trade secret, contractual safeguards, copyright, and patent measures supported the bulk of the agreements.

In accordance with previous studies, this study also looked at the exclusivity provisions of the licenses in this dataset to understand the extent

212. Cockburn, *supra* note 95, at 9 tbl.5.

213. *Id.* at 7.

214. 542/1,431 = 38%. A single agreement could transfer more than one type of right, for example, patent rights and trade secrets. In some cases, trade secrets appeared to be transferred in the absence of patent rights because, for example, patents were pending but had not been issued, or items had been specified in the agreement as "unpatented inventions." Among the agreements reviewed, one specifically referred to "unpatented" inventions; another mentioned inventions that were covered by patent applications that had not yet issued.

215. 1,261/1,431 = 88%.

to which intellectual property supported a contract's terms. In comparison to generally non–exclusive, "open source" software licensing agreements, the licenses studied were at times exclusive, but more frequently were non–exclusive or multi–exclusive, for example, by being exclusive in one territory or field of use, while non–exclusive in another.[216] Among all agreements, 34% had exclusive terms, 4% had non–exclusive terms, and 62% of the licenses were "multi–exclusive."[217]

The presence of patents or other forms of intellectual property[218] had ramifications for the amount of exclusivity. One of the arguments made in favor of intellectual property is that it provides a quantum of rights that can then be reduced or otherwise tailored by contract to fit the circumstances. The overwhelming majority of the software contracts (96%) fit this pattern, insofar as they contained some measure of exclusivity. However, it is also the case that intellectual property was not always needed to support this range of exclusivity options. Even when intellectual property was not a key component (N=558), non–exclusive and multi–exclusive rather than non–exclusive provisions predominated at almost the same rate as they did in intellectual property agreements. (Figure 7). Among these agreements, contract law appears to be doing much of the work in terms of allocating rights between parties.

V. CONCLUSION

Software innovation is transforming the U.S. economy. Yet, the paid market for software innovation is poorly understood, in part because of a lack of public information about the licensing and transfer of innovation between firms. This Article skirts these obstacles by drawing upon several proprietary datasets, exploring the market for software innovation through the lens of patent licenses and sales. This study finds that despite the intense academic and policy focus on software patent litigation, software patents are much more likely to be transferred than litigated (1.4–2.4% odds of being sold per year versus 1–2% odds of being litigated per lifetime), and argues that more attention should be paid to the market for innovation. Further, although legal decisions of the Supreme Court and new procedures have made it harder to enforce software patents, this study finds that the

216. David Jarczyk, *Dealing in Data*, INTELL. PROP. MAG. 44 (July/Aug. 2013), http://www.ktmine.com/wp-content/uploads/2014/04/044-045-IPM_July_August_2013-Feat.pdf.

217. 1,308 of the 1,431 software agreements had ascertainable exclusivity provisions. Of those 441 were exclusive, 809 were multi–exclusive, and 58 were non–exclusive.

218. This study specifically looked for copyright, trade secret, or trademark and related rights.

market for software innovation remains remarkably robust, with the number of software patents sold growing over 50% from 2012 to 2015. This development is attributable to the robustness of the demand for patents providing freedom to operate, the strength of software business models, and bargain shopping as the price of individual patents has gone down.

This Article distinguishes between transfers to support the transfer of technology as opposed to mere transfers of liability (generally through naked patent licenses). Contrary to other studies, this study finds that the majority of significant software patent agreements registered with the SEC (N=245) support true technology transfer. However, trade secrets and code were more important than patents for transferring software innovation between firms. In addition, it appears that large numbers of patents are being sold to avoid litigation or provide freedom to operate, not to access technology for development. The traditional narrative of patents enabling young companies to get access to the commercialization capabilities of larger, more established firms is not supported by the data—patents are two to three times more likely to go from an older company to a younger company, and from a higher revenue to lower revenue public company, based on available data.

These data support a nuanced, multi–dimensional role for software patents in the innovation ecosystem. The flexible, adaptable use of software patents in this ecosystem, sometimes to transfer technology, sometimes to transfer liability, and perhaps most often to protect the unbridled freedom to innovate, support the characterization of software patents as a currency of—rather than a tax on—innovation.

VI. APPENDIX

Figure A1: Distribution of Material Software Agreements Reported to the SEC Across Industries (2000–2015)[219]

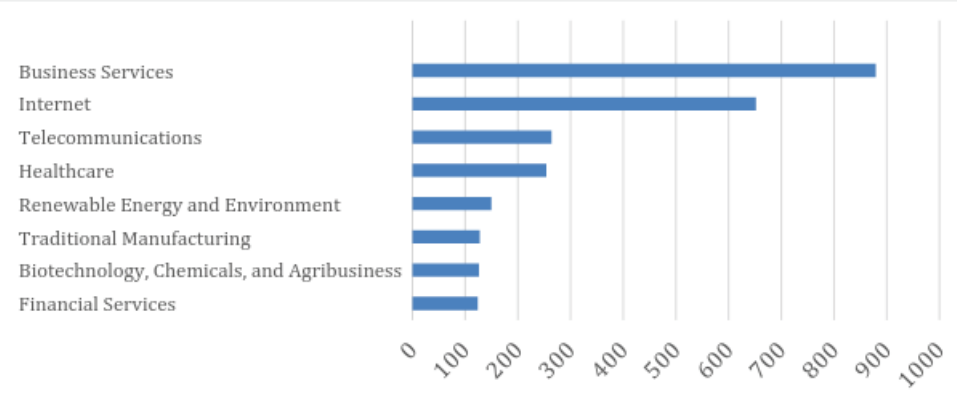
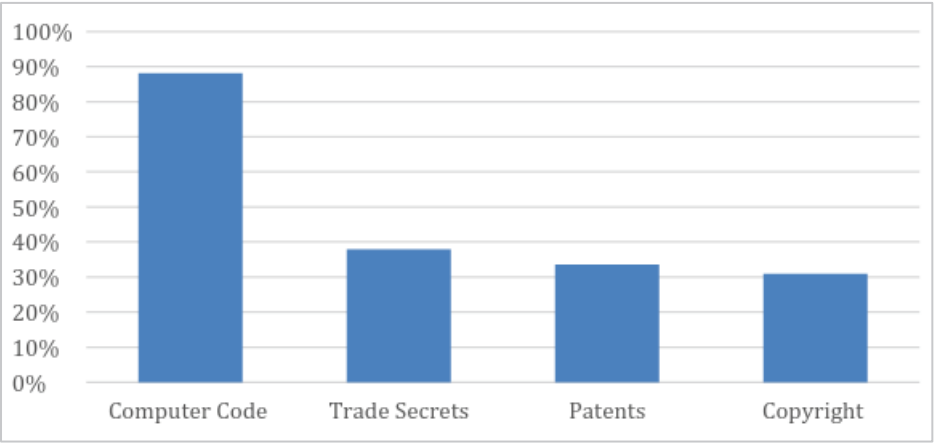


Figure A2: Share of Software Agreements in which Code, Trade Secrets, Patents, or Copyrights Were Considered Core



219. A single agreement may be assigned to one than one more industry.

Figure A3: Exclusivity Provisions Among Software Agreements (N=1431)