UNC SCHOOL OF LAW

NORTH CAROLINA LAW REVIEW

Volume 87
Number 5 *Frontiers in Empirical Patent Law Scholarship*

Article 7

6-1-2009

University Software Ownership and Litigation: A First Examination

Arti K. Rai

John R. Allison

Bhaven N. Sampat

UNIVERSITY SOFTWARE OWNERSHIP AND LITIGATION: A FIRST EXAMINATION*

ARTI K. RAI, JOHN R. ALLISON, AND BHAVEN N. SAMPAT**

Software patents and university-owned patents represent two of the most controversial intellectual property developments of the last twenty-five years. Despite this reality, and concerns that universities act as "patent trolls" when they assert software patents in litigation against successful commercializers, no scholar has systematically examined the ownership and litigation of university software patents. In this Article, we present the first such examination. Our empirical research reveals that software patents represent a significant and growing proportion of university patent holdings. Additionally, the most important determinant of the number of software patents a university owns is not its research and development ("R&D") expenditures (whether computer science-related or otherwise) but, rather, its tendency to seek patents in other areas. In other words, universities appear to take a "one size fits all" approach to patenting their inventions. This one size fits all approach is problematic given the empirical evidence that software is likely to follow a different commercialization path than other types of invention. Thus, it is perhaps not surprising that we see a number of lawsuits in which university software patents have been used not for purposes of fostering commercialization, but instead to extract rents in apparent holdup litigation. The Article

---

** Arti K. Rai is the Elvin R. Latty Professor at Duke University School of Law; John R. Allison is the Spence Centennial Professor of Business Administration at the McCombs School of Business, University of Texas, Austin; and Bhaven N. Sampat is Assistant Professor at Mailman School of Public Health, Columbia University. The authors gratefully acknowledge the support of the National Human Genome Research Institute and the U.S. Department of Energy. Earlier versions of this Article were presented at the University of Arizona Law School, University of North Carolina School of Law, the University of California, Berkeley (Boalt) School of Law, Columbia University School of Law, and the Washington University in St. Louis School of Law. We thank the participants in those symposia for their comments. We also thank Gerald Barnett, Jim Bessen, Richard Brandon, Robert Cook-Deegan, Rick Friedman, Bob Hunt, David Kappos, Ronald Mann, Robin Rasor, Mark Schankerman, and Manny Schecter for their comments. Thanks are also due to Matt Block, Nina Knierim, Erik Smith, and Cameron Westin for excellent research assistance. Finally, Colin Crossman provided invaluable help with the interviews.

concludes by examining whether this trend is likely to continue in the future, particularly given a 2006 Supreme Court decision that appears to diminish the holdup threat by recognizing the possibility of liability rules in patent suits, as well as recent case law that may call into question certain types of software patents.

INTRODUCTION

Software patents and university-owned patents represent two of the most controversial intellectual property developments of the past few decades. Various scholars have quarreled with the alleged vagueness and undue breadth of software patent claims.[1] Some have

1. *See, e.g.*, JAMES BESSEN & MICHAEL MEURER, PATENT FAILURE: HOW JUDGES, BUREAUCRATS, AND LAWYERS PUT INNOVATION AT RISK 187–214 (2008); Dan Burk & Mark Lemley, *Policy Levers in Patent Law*, 89 VA. L. REV. 1575, 1623 (2003) (arguing that the Court of Appeals for the Federal Circuit, which hears all appeals in patent cases, has an exaggerated sense of the skill of the ordinary computer scientist and is therefore likely to allow broad patents); Arti K. Rai, *Engaging Facts and Policy, A Multi-Institutional Approach to Patent System Reform*, 103 COLUM. L. REV. 1035, 1053–54 (2003) (arguing that because pure software patents are not limited to a particular physical machine or process, they may be problematic in terms of breadth). *But see* Robert P.

also suggested that, given the poor quality of prior art documentation and patent examiner training in the area of software, many issued software patents are likely to be obvious.[2] More generally, there is significant debate over the extent to which software patents are likely to foster (or hinder) innovation.[3] Because software products are often "complex" and may infringe many patents, some incumbent providers of information technology ("IT") products support limiting patentability.[4] Perhaps most notably, the U.S. Court of Appeals for the Federal Circuit (arguably prompted by the recent questioning of patents on "abstract ideas" by several members of the Supreme Court[5]) recently excluded processes from patentability (including

---

Merges, *Software and Patent Scope: A Report from the Middle Innings*, 85 TEX. L. REV. 1627, 1649–52 (2007) (arguing that some recent Federal Circuit cases have interpreted disclosure requirements for software patents more rigorously).

2. *See generally* Glynn Lunney, *E-Obviousness*, 7 MICH. TELECOMM. & TECH. L. REV. 363 (2001) (making this argument and further arguing that the Federal Circuit tends to affirm lower court determinations that a patent is nonobvious while reversing determinations that a patent is obvious and thus does not meet a prerequisite of patentability).

3. *Compare* James Bessen & Robert Hunt, *An Empirical Look at Software Patents*, 16 J. ECON. & MGMT. STRATEGY 157, 173–74, 180–85 (2007) (arguing that software patents are substitutes for research and development), *with* Ronald Mann, *Do Patents Facilitate Financing in the Software Industry*, 83 TEX. L. REV. 961, 999–1003 (2005) (arguing that software patents may help certain types of small software firms attract financing). Similarly, a study by Noel and Schankerman that is in tension with the findings of Bessen and Hunt examines software firms only. *See* Michael Noel & Mark Schankerman, *Strategic Patenting and Software Innovation* 12 (London Sch. of Econ. & Pol. Sci., Ctr. for Econ. Policy Research, Paper No. EI43, June 2006), *available at* http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1158320. These papers can be reconciled to the extent that Mann's argument in favor of patents is largely limited to small software firms, while Bessen and Hunt focus on software patents held by large firms both within and outside the software industry. For discussion of another empirical study on software patents, see *infra* note 144 and accompanying text.

4. These firms have supported legislative and judicial efforts to make patents easier to challenge and injunctive relief, particularly by nonmanufacturing entities, more difficult to secure. *See generally* eBay Inc. v. MercExchange, L.L.C., 547 U.S. 388 (2006) (clarifying that injunctive relief is not necessarily automatic upon a finding of infringement); Patent Reform Act of 2007, S. 1145, 110th Cong. § 321 (2007) (proposing a robust system of post-grant opposition proceedings). In contrast, while products in the biopharmaceutical industry may require many patented inputs for their creation, the products *themselves* are likely to be covered by only a few patents. Thus, one common strategy for avoiding patent thickets is secret infringement. *See* John Walsh et al., *Working Through the Patent Problem*, 299 SCIENCE 1021, 1021 (2003).

5. In *LabCorp v. Metabolite Lab, Inc.*, 548 U.S. 124, 124 (2006) (denying certiorari), Justices Breyer, Souter, and Stevens dissented from the dismissal of certiorari as improvidently granted, arguing that patents on abstract ideas and scientific principles should not be allowed and that the patent in question arguably covered a scientific principle. *Id.* at 127–28 (Breyer, J., dissenting).

software processes) that are not limited to a physical transformation or machine.[6]

In contrast with software patents, university-owned patents have existed for more than a few decades. The number of university-held patents has increased substantially, however, since the 1980 passage of the Bayh-Dole Act.[7] While the legal question was sometimes murky prior to 1980, Bayh-Dole made it unequivocally clear that universities can patent federally funded research.[8]

Assertive university patenting has attracted attention in both scholarly and popular literature.[9] Additionally, because universities, and sometimes even their exclusive licensees, are nonmanufacturing patentees, the intense debate over whether such patentees employ "holdup" strategies deleterious to innovation when they assert patents against successful commercializers directly implicates

---

6. *See In re* Bilski, 545 F.3d 943, 966 (Fed. Cir. 2008).

7. Bayh-Dole Act of 1980, Pub. L. No. 96-517, 94 Stat. 3017 (codified at 35 U.S.C. §§ 200–210 (2006)). *See* NAT'L SCI. BD., NAT'L SCI. FOUND., SCIENCE & ENGINEERING INDICATORS 2006, at 50–51 (2006), *available at* http://www.nsf.gov/statistics/seind06/pdf/volume1.pdf (noting that numbers of patents issued to academic institutions quadrupled from approximately 800 in 1988 to more than 3,200 in 2003).

8. Although not all university research is federally funded, the federal share represented sixty-four percent of university research and development ("R&D") in fiscal year 2004. *See* RONDA BRITT, NAT'L SCI. FOUND., ACADEMIC RESEARCH AND DEVELOPMENT EXPENDITURES: FISCAL YEAR 2004, at 8 (2006), *available at* http://www.nsf.gov/statistics/nsf06323/pdf/tab1.pdf. In electrical engineering and computer science ("EE/CS"), the federal percentage of university R&D in fiscal year 2004 was even higher, seventy-three percent and sixty-eight percent, respectively. *Id.* In contrast, industry accounts for only about five percent of university R&D (including EE/CS R&D). *Id.* These percentages have been fairly consistent over the last twenty years. *Id.* For example, from 1996 to 1998, the years probably most relevant to patents issued in 2002, the federal funding percentage for EE/CS was seventy percent, sixty-eight percent, and sixty-seven percent, in each respective year. *Id.* The overall federal funding percentage during those years was about sixty percent. *Id.*

We report these statistics to buttress our supposition that most software-related research done by universities is federally (or at least publicly) funded and thus subject to the types of economic analysis usually employed in thinking about publicly funded research. Unfortunately, although Bayh-Dole requires universities to report federal funding when they file for patents, many do not. *See* University of California, The Bayh-Dole Act: A Guide to the Law and Implementing Regulations, http://www.ucop.edu/ott/faculty/bayh.html (last visited Apr. 21, 2009). Thus, we cannot rely on such reports to determine the funding source of research that led to specific software patents.

9. In a popular account, Jennifer Washburn suggests that universities are being corrupted by their interest in commercial activities, including patenting. *See generally* JENNIFER WASHBURN, UNIVERSITY, INC.: THE CORPORATE CORRUPTION OF AMERICAN HIGHER EDUCATION (2005) (discussing universities and their interest in commercial activities). For a comprehensive analysis of available data on the impact of Bayh-Dole, see generally DAVID C. MOWERY ET AL., IVORY TOWER AND INDUSTRIAL INNOVATION (2004).

universities.[10] That said, almost all analyses of university patenting have focused on patenting within the life sciences.[11] This focus is perhaps not surprising, as the major economic argument put forward in the legislative history of the Bayh-Dole Act—that patents on publicly funded invention promote commercialization of such invention[12]—would appear to apply most clearly to areas of the life sciences, such as drug development. Conventional wisdom suggests that, without the quasi-monopoly protection of a patent on the small molecule chemical, few firms would be interested in taking a potentially promising drug candidate through the expensive clinical trial and approval process.[13]

In the case of publicly funded software, by contrast, the need for a patent and exclusive license to promote further development is less apparent.[14] Although development costs are not uniformly low, they

10. *See generally* Mark Lemley, *Are Universities Patent Trolls?*, 18 FORDHAM INTELL. PROP. MEDIA & ENT. L.J. 611 (2008) (providing a nonempirical analysis of whether universities should be considered patent "trolls"). The most celebrated recent example of a lucrative holdup strategy was that pursued by a patent holding company, NTP, against Research in Motion ("RIM"), the maker of the BlackBerry. In that case, RIM settled for $612.5 million in order to avoid the imminent threat of court-ordered injunctive relief, even though the validity of several of the asserted patents had been called into question by the Patent and Trademark Office. *See* Yuki Noguchi, *BlackBerry Patent Dispute is Settled*, WASH. POST, Mar. 4, 2006, at A1. For discussion of holdup litigation by universities, see *infra* Part IV.B.

11. *See, e.g.*, Pierre Azoulay, Ryan Michigan & Bhaven N. Sampat, *The Anatomy of Medical School Patenting*, 357 NEW. ENG. J. MED. 2049, 2049–56 (2007); Pierre Azoulay, Waverly Ding & Toby E. Stuart, *The Determinants of Faculty Patenting Behavior: Demographics or Opportunities?*, 63 J. ECON. BEHAV. & ORG. 599, 599 (2007); Fiona Murray & Scott Stern, *Do Formal Intellectual Property Rights Hinder the Free Flow of Scientific Knowledge: An Empirical Test of the Anti-Commons Hypothesis*, 63 J. ECON. BEHAV. & ORG. 648, 648 (2007).

12. Although the text of Bayh-Dole does not suggest a particular licensing model, the legislative history of the bill that eventually became the Bayh-Dole Act, as well as that of similar bills that were being discussed at the time, indicates a focus on exclusive licenses. House Report 96-1307 notes the importance of exclusive licensing for attracting capital necessary for development. H.R. REP. NO. 96-1307, pt. 1, at 3, 5 (1980). Senate Report 96-480 asserts that, because nonexclusive licenses were generally viewed dismissively in the business community "as no patent protection at all," nonexclusive licensing by the government had not been successful in producing development. S. REP. NO. 96-480, at 28 (1979).

13. The empirical evidence also indicates that patents are more important for recouping R&D investment in the pharmaceutical industry than in other manufacturing industries. *See* Wesley Cohen, Richard R. Nelson & John P. Walsh, *Protecting Their Intellectual Assets: Appropriability Conditions and Why U.S. Manufacturing Firms Patent (or Not)* 2 (Nat'l Bureau of Econ. Research, Working Paper No. 7552, Feb. 2000), *available at* http://www.nber.org/papers/w7552.pdf (discussing the recouping of R&D costs in the context of the pharmaceutical industry and other manufacturing industries).

14. Even in the life sciences, the availability of patents on improvements, as well as the presence of absorptive capacity in commercial firms, may diminish the need for

are likely to be low relative to those in the biopharmaceutical industry.[15] Indeed, in certain cases of open source software development, firms derive revenue not from property rights over the software product itself, but from a strategy that monetizes the value of support services and complementary hardware.[16] Thus, if universities are in fact making strong proprietary claims on software, scholars should be concerned. The social welfare argument for such claims is more tenuous than in the life sciences.

Despite IT sector complaints about university behavior,[17] as well as prominent lawsuits involving software patents,[18] the subject of university software ownership and litigation has not been studied systematically. Indeed, this Article represents the first systematic study of which we are aware.[19] We rely in part on a unique, hand-curated database of university software patents.[20] This quantitative analysis is supplemented by interviews conducted with technology transfer officers at those universities that own large numbers of software patents, as well as academic scientists and other officials prominent in the open source movement.

---

exclusive licensing on certain inventions, such as research tools. MOWERY ET AL., *supra* note 9, at 158 (discussing the manner in which Columbia's DNA co-transformation technology was developed commercially without the need for exclusive licensing).

15. *See* Joseph A. DiMasi et al., *The Price of Innovation: New Estimates of Drug Development Costs*, 22 J. HEALTH ECON. 151, 166 (2003) (giving an estimate of $802 million for new chemical molecules developed entirely in the private sector). This estimate, which includes an opportunity cost of capital of about $400 million, would presumably be somewhat lower where some of the relevant work had already been done in a federally funded institution. *See id.*

16. *See generally* Andrea Bonaccorsi & Cristina Rossi, *Why Open Source Software Can Succeed*, 32 RES. POL'Y 1243 (2003) (discussing examples ranging from software publishers like Red Hat to hardware manufactures like IBM).

17. *See, e.g.*, *Hearing on "Bayh-Dole—The Next 25 Years" Before the H. Comm. on Science and Technology*, 110th Cong. 46–56 (2007) (statement of Wayne Johnson, Vice-President, University Relations Worldwide, Hewlett-Packard); Steve Lohr, *IBM and Universities Plan Collaboration*, N.Y. TIMES, Dec. 14, 2006, at C11 (noting a statement by an IBM vice president that "[u]niversities have made life increasingly difficult to do research . . . because of all the contractual issues around intellectual property").

18. *See, e.g.*, Eolas Techs. Inc. v. Microsoft Corp., 399 F.3d 1325 (Fed. Cir. 2005).

19. Ajay Agrawal and Rebecca Henderson touch on a related subject in their examination of the patenting practices of the MIT EE/CS faculty. Ajay Agrawal & Rebecca Henderson, *Putting Patents in Context: Exploring Knowledge Transfer from MIT*, 48 MGMT. SCI. 44, 44 (2002). Based on their research, they conclude that patenting is a "minority activity" for most faculty members in the EE/CS department (and the mechanical engineering department). *Id.*

20. Our quantitative analysis focuses on patents primarily because there is no comprehensive data on the extent to which copyright is asserted by universities. Copyright attaches as soon as the software is created. Because there is no need to register copyrights, it is difficult to know the total volume of university software protected by copyright.

The combination of our quantitative and qualitative inquiry yields a number of important results. First, software patents represent a significant and growing percentage of university patent holdings.[21] Second, university software patenting practices tend to mimic their nonsoftware patenting practices. The data suggests that those universities that have a higher patent propensity in general are also more likely to obtain software patents.[22] Similarly, our interviews show that some universities view software as similar to other, more physical inventions.[23] The difficulty with this view is that software is likely to follow a different commercialization path than other inventions. Thus, it is perhaps not surprising that we see a fair number of litigated cases involving software patents, and that almost all of these appear to represent situations where the university and/or its exclusive licensee is asserting the patent against an entity that has successfully commercialized software independent of the patent.[24] Notably, in the majority of these cases, the university's argument has lost on grounds of either patent invalidity or noninfringement.[25]

The main rationale for supporting patenting would, therefore, appear to be the promotion of start-up businesses, presumably on the theory that start-ups, and market-based activities more generally, are likely to be more innovative than activities in large, vertically-integrated incumbent firms.[26] However, whether patents are necessary for start-up promotion is not as clear in the software area as it is, for example, in the biotechnology industry. Moreover, in contrast with biotechnology, where copyright is not available, universities can use software copyright to achieve revenue generation goals.[27]

It may be that university software patenting is a transient phenomenon. As we discuss below,[28] some technology transfer officers argue that they no longer view the patenting of software as a particularly good mechanism for technology transfer.[29] In addition, the 2006 Supreme Court decision in *eBay Inc. v. MercExchange, L.L.C.*,[30] allowing district courts discretion to award damages even

21. *See infra* Part II.A.
22. *See infra* Part II.B.
23. *See infra* Part III.
24. *See* discussion *infra* Part IV.B.
25. *See* discussion *infra* Part IV.B.
26. *See* discussion *infra* Part IV.B.
27. *See* discussion *infra* Part V.B.
28. *See infra* Part V.C.
29. *See infra* Part V.C.
30. 547 U.S. 388 (2006).

after validity and infringement have been found, coupled with recent Federal Circuit case law that may call into question certain types of software patents, could make software patenting and litigation less attractive to universities in the future.

This Article proceeds as follows. Part I discusses the history of software patenting and how it influenced our methodology for compiling a database of university software patents. Next, Part II presents our quantitative results; specifically, we identify trends in, and determinants of, university software patenting. Part III integrates these quantitative results with results from our interviews. Part IV then discusses the "holdup" features present in much university litigation over software patents. More generally, it provides a policy analysis of why universities' apparent "one size fits all" approach to patenting is problematic. Finally, Part V discusses the path forward, with a focus on whether we are likely to see more of this unitary approach (and concomitant litigation) in the future.

I. SOFTWARE PATENTING: HISTORY AND METHODS OF IDENTIFYING

To identify university patents, we began by using the U.S. Patent and Trademark Office's ("PTO") Cassis database to identify all patents issued in 1982, 1987, 1992, 1997, and 2002 that were assigned to institutions classified as Research or Doctoral Universities in the Carnegie Commission of Higher Education's 1972 or 1994 reports. We chose these particular years because they span a series of shifts in the legal regime surrounding software produced at universities.[31] Not only did university patenting increase significantly after the 1980 passage of the Bayh-Dole Act,[32] but patent jurisprudence in the area of software also evolved considerably. Because this evolution is closely related to the manner in which we define the term "software patent," we describe it in some detail below.

31. A methodology that sampled one year of filing would have tracked more precisely the impact of particular software cases on filing behavior. However, because we did not aim to measure the precise impact of specific cases, sampling by year of issue was sufficient for purposes of getting a general sense of trends. Sampling by year of issuance was a more important issue for our regression analyses, which require that we assess patenting against (inter alia) R&D expenditures. We address this issue below. *See infra* note 70 and accompanying text.

32. *See supra* note 7 and accompanying text.

A. *The History of Software Patents*

In the 1970s, the dominant intellectual property regime for software was copyright, not patent.[33] A 1972 Supreme Court case, *Gottschalk v. Benson*,[34] appeared to reject software (in that case, a computerized method for converting decimal numbers to binary numbers) as patentable subject matter on the grounds that patent law did not encompass abstract scientific or mathematical principles.[35] Further, several years later, in the 1980 amendments to the Copyright Act of 1976, Congress expressly endorsed copyright as an appropriate protection regime for software.[36]

The intellectual property terrain shifted in the 1980s. In the 1981 decision *Diamond v. Diehr*,[37] the Supreme Court gave its first clear indication that certain types of software-implemented inventions were patentable. *Diehr* narrowed *Gottschalk* by upholding as patentable subject matter a rubber-curing process that used software to calculate cure time. According to the *Diehr* Court, the physical transformation of the rubber "into a different state or thing" took the invention being claimed out of the realm of abstraction.[38]

Through the 1980s, the Court of Appeals for the Federal Circuit generally followed a test somewhat similar to that enunciated in *Diehr*. Under this test, if an invention's claims involved nothing more than an algorithm, then the invention could not be patented.[39]

---

33. *See generally* Peter S. Menell, *An Analysis of the Scope of Copyright Protection for Application Programs*, 41 STAN. L. REV. 1045 (1989) (analyzing various means of legal protection for computer technologies, including copyright and patent law, and their scope of protection). To be sure, some software patents may have been issued in disguised form in the 1970s. *See* E-mail from Richard Brandon, Assistant Gen. Counsel, Univ. of Mich. Office of Tech. Transfer, to Arti K. Rai, Professor of Law, Duke Univ. Sch. of Law (Nov. 21, 2006 17:19 EST) (on file with the North Carolina Law Review). This fact is not relevant for our purposes.

34. 409 U.S. 63 (1972).

35. Although *Gottschalk* is generally considered a subject matter case, the Court may also have been concerned with breadth—the patent in question was not restricted to any particular implementation of the algorithm. *Gottschalk*, 409 U.S. at 65–72. In general, as noted above, pure software patents of the type at issue in *Gottschalk* may be broader than patents covering software restricted to a particular physical process or machine. *See* Rai, *supra* note 1, at 1104–05.

36. Act of Dec. 12, 1980, Pub. L. No. 96-517, § 10, 94 Stat. 3105, 3028 (codified as amended at 17 U.S.C. §§ 101, 117 (2006)).

37. 450 U.S. 175 (1981).

38. *Id.* at 175. In fact, the *Diehr* Court argued that it had long viewed "[t]ransformation and reduction of an article 'to a different state or thing' [as] *the* clue to the patentability of a process claim that does not include particular machines." *Id.* at 184 (emphasis added) (citations omitted).

39. *See* Arrhythmia Research Tech., Inc. v. Corazonix Corp., 958 F.2d 1053, 1058 (Fed. Cir. 1992).

However, if the claims involved a mathematical algorithm that was "applied to, or limited by, physical elements or process steps," such claims would constitute patentable subject matter.[40] The overall message to patent attorneys was that software could be patented, but it had to be claimed as something else. Assuming an approximate examination pendency of two years during the relevant period,[41] patents issued in our sample years of 1987 and 1992 should reflect patenting of software as "something else."

Moreover, as the patent option was becoming more attractive, copyright was becoming much less so. In the early 1990s, a series of appellate court decisions made it clear that copyright covered primarily the literal source code of the program.[42]

Greater changes lay in store. In the 1994 case of *In re Alappat*,[43] the Federal Circuit effectively eliminated any limitation on patenting software by concluding that subject matter criteria could be met by claiming software as a new machine—a "special purpose" computer—when it was executed.[44] Presumably all software could produce such a special purpose computer and hence be patentable. Again, assuming a two-year examination pendency, patents issued in 1997 should begin to incorporate any effects that decisions like *Alappat* had on filing incentives. Also, in 1996, the PTO issued software guidelines that broadly allowed software as patentable subject matter whether the software was claimed as a machine or a process.[45] Two years later, the Federal Circuit's decision in *State Street v. Signature Financial Group*[46] similarly rejected any special subject matter test for software, finding that software (like all inventions) is patentable if it produces a "useful" result.[47] After *State Street*, there was no need for even the fig

40. *Id.*

41. The PTO calculates that average pendency was about two years over the period from 1988 to 2000. *See* John L. King, *Patent Examination Procedures and Patent Quality*, *in* PATENTS IN THE KNOWLEDGE-BASED ECONOMY 54, 63 (Wesley M. Cohen & Stephen A. Merrill eds., 2003) (citing data from Annual Reports of the PTO).

42. *See, e.g.*, Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807, 817 (1st Cir. 1995), *aff'd*, 516 U.S. 233 (1996); Computer Ass'n Int'l v. Altai, Inc., 982 F.2d 693, 702 (2d Cir. 1992).

43. 33 F.3d 1526, 1545 (Fed. Cir. 1994). The *Alappat* decision was anticipated to some extent by cases like *In re Iwahashi*, 888 F.2d 1370, 1375 (Fed. Cir. 1989) (holding that just because an apparatus is operated as an algorithm does not make it non-statutory).

44. *Alappat*, 33 F.3d at 1545.

45. *See* Examination Guidelines for Computer-Related Inventions, 61 Fed. Reg. 7,478, 7,478 (Feb. 28, 1996) (outlining how PTO examiners should scrutinize computer-related inventions).

46. 149 F.3d 1368 (Fed. Cir. 1998).

47. *Id.* at 1375 (arguing that the question of whether a claim encompasses patentable subject matter should focus on "practical utility").

leaf of a physical machine or process. Thus, patents issued in 2002 should incorporate any impact of the PTO guidelines and the *State Street* decision on filing incentives.[48]

B. *Identifying University Software Patents*

Given this history, it is perhaps not surprising that identifying "software" patents is very difficult. Even now, there is no universally accepted definition of what a software patent is. To our knowledge, there have been only a few significant efforts to identify a large data set of software patents.[49] An initial paper by Stuart Graham and David Mowery,[50] which does not attempt to define the term "software patent," relies upon certain International Patent Classifications ("IPCs")[51] as limited to patents in those classes owned by large software firms.[52] A more recent paper by Graham and Mowery uses particular U.S. patent classes, once again limited to patents in those classes which are owned by large software firms.[53] The Graham and Mowery approach is not likely to be significantly overinclusive so long as it is limited to patents owned by packaged software firms. However, their approach may be quite underinclusive, missing

48. As noted earlier, the pendulum has arguably begun to swing back. The recent Federal Circuit decision in *In re Bilski* states that patent claims to a process must be tied to a physical transformation or machine. *See supra* note 6 and accompanying text. This very recent decision is not, however, directly relevant to our empirical analysis. We discuss the implications of *Bilski* for enforcement of university software patents *infra* Part V.A.

49. We exclude from our discussion a paper by Iain Cockburn and Meghan MacGarvie that focuses on patents held by firms in twenty-seven specific software markets. *See* Iain M. Cockburn & Megan J. MacGarvie, *Entry, Exit and Patenting in the Software Industry* 12 (Nat'l Bureau of Econ. Research, Working Paper No. 12563, Oct. 2006), *available at* http://www.nber.org/papers/w12563 (finding that firms are less likely to enter software markets in which there are more software patents but that, all else equal, firms that have software patents are more likely to enter these markets).

50. Stuart J.H. Graham & David C. Mowery, *Intellectual Property Protection in the U.S. Software Industry*, *in* PATENTS IN THE KNOWLEDGE-BASED ECONOMY 219, 231 (Wesley M. Cohen & Stephen A. Merrill eds., 2003).

51. Graham and Mowery use IPC classes G06F (subclasses 3, 5, 7, 9, 11, 12, 13, and 15), G06K (subclasses 9 and 15), and H04L (subclass 9). *Id.*

52. *See id.* at 232 n.37 (noting that analysis focuses on patents assigned to large, publicly traded software firms).

53. Stuart J.H. Graham & David C. Mowery, *Software Patents: Good News or Bad News?*, *in* INTELLECTUAL PROPERTY RIGHTS IN FRONTIER INDUSTRIES: SOFTWARE AND BIOTECHNOLOGY 45, 56–57 (Robert Hahn ed., 2005). The U.S. patent classes are 345, 358, 382, 704, 707, 709, 710, 711, 713, 714, 715, and 717. Similarly, Bronwyn Hall and Meghan MacGarvie rely on PTO classes. They do not, however, limit their use of these classes to packaged software firms. *See* Bronwyn H. Hall & Meghan MacGarvie, *The Private Value of Software Patents* 16–19 (Nat'l Bureau of Econ. Research, Working Paper No. 12,195, May 2006), *available at* http://www.nber.org/papers/w12195 (discussing data sets that are not confined to packaged software firms).

software patents assigned to other firms as well as software patents in other patent classes.[54]

Another significant effort to identify a large set of software patents, by James Bessen and Bob Hunt,[55] defines "software patent" to include patents on inventions in which the data processing algorithms are carried out by code either stored on a magnetic storage medium or embedded in chips ("firmware").[56] Rejecting the

54. To get a sense of false positives and false negatives, we assessed how the two Graham and Mowery approaches classified the 2,942 university patents issued in 2002. As shown in the Appendix Tables A1 and A2, very few of the patents classified by us as nonsoftware were classified as software by either the GM-IPC approach or the GM-PTO approach. However, the GM-IPC approach did not classify as software eighty-six percent of the patents we classified as software. Similarly, the GM-PTO approach did not classify as software eighty-two percent of the patents we classified as software. An analysis conducted by Bronwyn Hall and Meghan MacGarvie compared Graham-Mowery with earlier datasets manually compiled by Allison and determined that an approach that uses patent classifications misses about fifty percent of software patents. *See* Hall & MacGarvie, *supra* note 53, at 17 (finding mixed effects of changes in legal doctrine on market value and stock returns of software firms).

Of course, as the analysis by Hall and MacGarvie suggests, *see* Table A2 Appendix *infra*, even significant percentages of Type I and Type II errors do not necessarily have a systematic impact on regression results. In our case, however, since the number of patents in our data set was tractable, and we needed to do manual work to determine departmental origin in any event, *see infra* Part II.C, we preferred a more precise approach.

55. *See* Bessen & Hunt, *supra* note 3.

56. As Bessen and Hunt note, one of the current authors, John Allison, earlier employed a definition of software patent that excluded firmware, including only inventions in which the code implementing the data processing algorithms are stored on a magnetic storage medium. *See* John R. Allison & Mark Lemley, *Who's Patenting What? An Empirical Exploration of Patent Prosecution*, 53 VAND. L. REV. 2099, 2110 (2000). Allison also employed this definition in later articles. *See* John Allison & Mark Lemley, *The Growing Complexity of the Patent System*, 82 B.U. L. REV. 77 (2002) (discussing patent trends over the last thirty years); John Allison & Emerson Tiller, *The Business Method Patent Myth*, 18 BERKELEY TECH. L.J. 987 (2003) (arguing that business method patents should not receive special treatment by the PTO). The reasons for using this definition were a combination of initial doubt and compromise with a co-author, followed by a need for consistency. Each of these articles made use of the same data set of 1,000 randomly selected patents issued between mid-1996 and mid-1998. After a great deal more experience gained from closely reading thousands of computer-related patents, Allison became convinced that the definition should include firmware. When he used the same set of 1,000 randomly selected patents in a subsequent article, he studied each patent again and reclassified them using a definition that included firmware. *See generally* John R. Allison et al., *Valuable Patents*, 92 GEO. L.J. 435 (2004) (discussing characteristics of litigated patents). Allison has used this more inclusive definition in studying the approximately 20,000 patents issued during 1998–2002 to almost 1,000 firms appearing in the Software 500 list in those years. This list ranks firms according to their gross revenues in software and services and includes many firms that are primarily manufacturers in addition to firms that produce only software. *See* John R. Allison et al., *Software Patents, Incumbents, and Entry*, 85 TEX. L. REV. 1579, 1594–95 (2007) (detailing methodology). *See generally* John R. Allison & Ronald J. Mann, *The Disputed Quality of Software*

use of patent classifications,[57] Bessen and Hunt study a random sample of patents and classify them according to their definition. Using characteristics of patents Bessen and Hunt find to fit their definition, they then develop a keyword search algorithm to identify software patents.[58]

Although the Bessen and Hunt definition of software patent is reasonable, there are pitfalls associated with using automated keyword searches to identify such patents. From one of the co-author's (John Allison's) study of thousands of computer-related patents, it is clear that the use of language in the titles, abstracts, written descriptions, and claims of patents, even in those dealing with the same area of technology, can be highly idiosyncratic among different patent owners. In general, a common criticism of patents in the IT industries is that the industries lack a standardized vocabulary for claims.[59] Moreover, software is a critical part of inventions in so many fields that reliance on particular search terms could produce a data set that has both false positives and false negatives.[60]

Patents, 85 WASH. U. L. REV. 297 (2007) (analyzing firm-held patents and their characteristics).

57. *See* Bessen & Hunt, *supra* note 3, at 163–64 (discussing methodology for identifying software patents). The Bessen and Hunt definition of a software patent appears to include patents on inventions that "use" software as part of the invention but excludes those that "use" off-the-shelf software:

> Our concept of software patent involves a logic algorithm for processing data that is implemented via stored instructions; that is, the logic is not "hard-wired." These instructions could reside on a disk or other storage medium or they could be stored in "firmware," that is, a read-only memory, as is typical of embedded software. But we want to exclude inventions that do not use software as part of the invention. For example, some patents reference off-the-shelf software used to determine key parameters of the invention; such uses do not make the patent a software patent.

Id. at 163.

58. The keyword search algorithm initially identifies a set of patents that use the words "software," "computer," or "program" in the claims or specification. *Id.* at 185. Patents within the set that contain the word "semiconductor," "chip," "circuit," "circuitry," or "bus" are then excluded, as well as patents that contain the word "antigen," "antigenic," or "chromatography." *Id.*

59. *See, e.g.*, BESSEN & MEURER, *supra* note 1.

60. *See generally* Bessen & Hunt, *supra* note 3 (identifying substantial degrees of over and under inclusiveness in the data set generated by their keyword search). Table A3 in the Appendix uses university patents issued in 2002 to compare the Bessen and Hunt ("BH") approach with our own approach. The two approaches yield an approximately comparable number of total patents (396 patents using our approach versus 415 using the BH approach). However, fifty-one percent of the patents our approach identifies as software are not identified as such by the BH algorithm. Moreover, we classify as nonsoftware fifty-three percent of the patents that BH classify as software. Similarly, one recent study that uses software experts to read a sample of the BH patents asserts that

Instead of relying upon prior approaches, we formulated a definition of "software patent" based on the extensive experience one of us (Allison) has had in reading software patents. Our definition of software patent is "a patent in which at least one claim element consists of data processing—the actual manipulation of data—regardless of whether the code carrying out that data processing is on a magnetic storage medium or embedded in a chip." Not only is it possible to apply the definition consistently, but it also captures the realities of claim drafting. In Allison's experience, patent claims often include the prior art, with only one or two elements covering the purportedly novel and nonobvious advance. For example, a claim may read as though it covers a generic router, printer, magnetic resonance imaging machine, or other hardware, when in fact the only purported novelty is in one element consisting of a function carried out by algorithms. Some of this may be a consequence of the fact that, prior to *Alappat* and the 1996 PTO guidelines, software had to be claimed not as a new algorithm *per se* but instead as a new machine that allegedly did something different because of the new algorithm.[61] More fundamentally, a claim covers the entire invention, and in a case like this, the entire invention is not just the new algorithm in isolation; rather, it is a piece of hardware that allegedly does something different because of the new algorithms.

In a large set of patents, it is a practical impossibility to include only those patents in which the software element (as contrasted with other elements of the invention) is novel and nonobvious. In order to restrict ourselves to patents in which the software element was novel and nonobvious, we would need to employ a person having ordinary skill in the art to conduct a very thorough study of the relevant prior art. Even then, the question would be plagued by doubt because issues of novelty and nonobviousness are typically difficult to resolve even after a full evidentiary exploration in court. But the fact that, under our definition, the data processing must be identified in a claim element does suggest that software is sufficiently important to novelty and nonobviousness for the patent claim drafter to include it as a limitation of the claim (thereby narrowing the claim). In other words, it would be foolhardy for a patent attorney to include such an element unless it was a critical part of the invention.

---

more than fifty percent represented Type II errors. *See* Anne Layne-Farrar, *Defining Software Patents: A Research Field Guide* 1–21 (AEI-Brookings Joint Ctr., Working Paper No. 05-14, Aug. 2005), *available at* http://aei-brookings.org/admin/authorpdfs/redirect-safely.php?fname=../pdffiles/phpAW.pdf.

61. *See supra* notes 43–45 and accompanying text.

Our approach does have limits. One problem with our approach is that it involves the slow and laborious process of reading patents. Although the decision on many patents is clear, there will always be a substantial percentage that must be studied with great care.[62] Moreover, a degree of subjective judgment is occasionally required. However, at least in the case of a relatively small data set, we believe that increased accuracy more than compensates for time intensity and the absence of algorithmic criteria readily replicable by automated methods. We do not claim that our data set of university-owned software patents is perfect, but we do contend that our error rate is very small—certainly smaller than in any data set acquired by means of patent classifications or keyword searches.

In addition to identifying which patents out of the more than 7,600 university-owned patents in our sample are software patents, we also identified a subset of those that may be called "pure software patents." These are patents in which the claims consist only of data processing—that is, the entire invention consists of algorithms.[63] This task required thorough study of each of the patents that had already been identified as a software patent. Although the process of identifying "pure" software patents was accomplished with a high degree of accuracy, there was a small number about which reasonable minds could differ. Thus, for this second stage, we also do not profess to have achieved perfection, but we do maintain that our error rate is very low.

62. However, if one is studying a large population of patents from the computer-related industries, the percentage that must be carefully scrutinized is far higher than if one is studying a population of patents across a broad array of fields (as in this Article).

63. These sorts of patents could presumably issue with any frequency only after the Federal Circuit's 1994 *Alappat* decision. However, as discussed further below, *see infra* Part V.A, even these pure software claims typically involve the use of a computer and, in some cases, a trivial nondata processing element such as a generic input, output, or storage element. Thus, they may survive even after *In re Bilski*.

II. UNIVERSITY SOFTWARE PATENTING: TRENDS, DETERMINANTS, AND DEPARTMENTAL ORIGIN

A. Trends

Figure 1: University Software Patents, By Type and Year

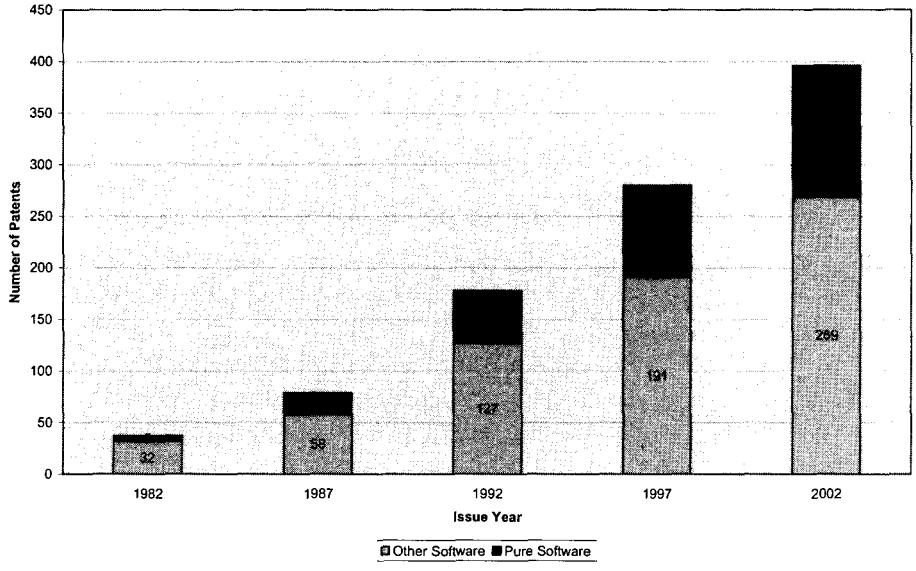

Figure 1 shows that university software patenting increased more than ten-fold over the 1982–2002 period, from thirty-seven patents in 1982 to 396 patents in 2002. Over this period, the "pure software" proportion of university software patents also increased dramatically, from thirteen percent to thirty-two percent of all university software patents. The latter change is hardly surprising. While the patentability of pure software was unclear in the 1980s, its status became much more secure in the 1990s.[64] Over these two decades, university software patenting also grew at a faster rate than university patenting overall. As a consequence, the software share of university patents rose from nine percent in 1982 to thirteen percent in 2002, as seen in Figure 2.

64. *See supra* notes 42–45 and accompanying text.

Figure 2: University Software Patents as a Share of All University Patents

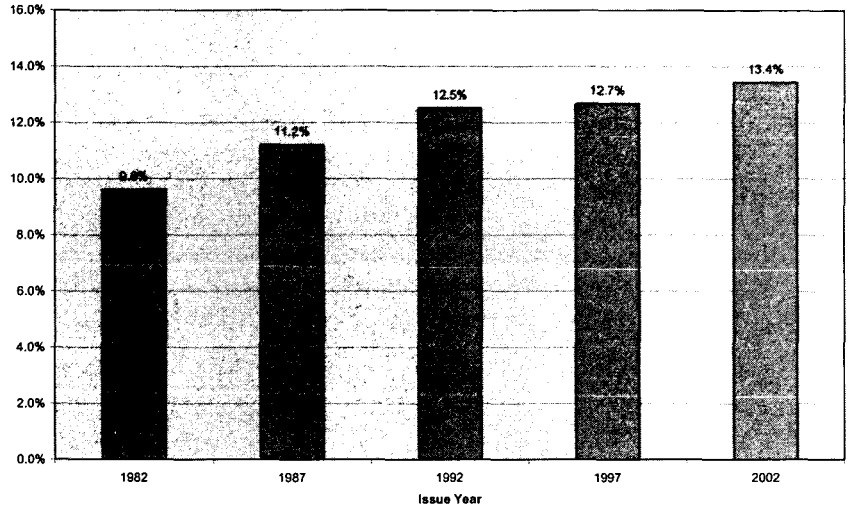

Table 1 below lists the fifteen universities that received the most software patents in 2002.[65] Together, these fifteen institutions accounted for sixty percent of all university software patents issued in 2002. The top five institutions alone—the Massachusetts Institute of Technology ("MIT"), the University of California, Stanford University ("Stanford"), the California Institute of Technology ("Caltech"), and the University of Texas—accounted for over a third (34.2%) of all university software patents. The top five patentees also represented the top five university patentees overall in 2002. However, moving further down the list of the top fifteen, we see that a number of the top software patentees are not among the top university patentees overall. The University of Washington (sixth in software patenting/fifteenth in overall patenting), the Georgia Institute of Technology ("Georgia Tech") (eighth/twentieth), Carnegie Mellon (ninth/fifty-first), the University of Rochester (twelfth/fiftieth) and the University of Illinois (fourteenth/twenty-eighth) particularly stand out as institutions substantially more prominent in software patenting than overall patenting.

65. To be sure, the 2002 data may be somewhat unusual in that it reflects patent filings that occurred during the "dot-com" bubble of the late 1990s. However, with a few exceptions, these universities also received the largest number of software patents over the course of the sampling.

Table 1: University Software Patenting, Overall Patenting, and Pure Software Patenting in 2002

	Rank in Patenting (Issue Year 2002)		
University	**Software**	**Overall**	**Pure Software**
Massachusetts Institute of Technology	1	2	1
University of California	2	1	2
Stanford University	3	4	4
California Institute of Technology	4	3	23
University of Texas	5	5	41
University of Washington	6	15	6
University of Wisconsin	7	7	8
Georgia Institute of Technology	8	20	3
Carnegie Mellon University	9	51	13
Johns Hopkins University	10	6	11
State University of New York	11	8	20
University of Rochester	12	50	14
University of Pennsylvania	13	13	42
University of Illinois	14	28	5
Columbia University	15	14	10

With respect to the patenting of "pure" software, Table 1 shows that the top three patenting institutions overall also ranked among the top recipients of pure software patents (first, second, and fourth). In contrast, although Caltech and the University of Texas ranked high in overall software patenting (fourth and fifth, respectively), they ranked relatively low in the patenting of pure software (twenty-third and forty-first, respectively). The University of Washington, Georgia Tech, Carnegie Mellon, the University of Rochester, and the University of Illinois—mentioned earlier as standing out in software patenting relative to overall patenting—also stand out with respect to numbers of pure software patents (sixth, third, thirteenth, fourteenth, and eleventh, respectively).

As these examples suggest, several factors may affect university software patenting. First, the amount of software-related R&D, and thus the output of software, may matter. Second, the size of the overall research enterprise may matter, because software can be

developed in many parts of the university. Third, an individual university's nonsoftware-related propensity to patent, i.e., the share of nonsoftware research outputs it patents (either because its researchers are prone to file invention disclosure statements and push for patents on those disclosures or because technology transfer officers are prone to seek patents on disclosures), may also affect software patenting.[66] Although there are undoubtedly other factors that affect software patenting—for example, how software ownership in particular is viewed in the technology licensing office or among faculty[67]—these factors are difficult to measure quantitatively. We examine some of these factors in our qualitative analyses below.

B. Patent Production Functions

To examine the relationship of software R&D, overall R&D, and nonsoftware patent propensity to university level software patenting, we collected data on R&D funds as well as nonsoftware patenting for all of the 202 Carnegie research universities in our sample. The unit of analysis for the "patent production functions" we estimate is a university in a given issue year. For each institution, we created variables measuring the number of software patents and number of other patents issued in 1982, 1987, 1992, 1997, and 2002. We used the National Science Foundation's WebCaspar database[68] to collect R&D data for these universities for the 1977–2002 time period.[69] Specifically, we collected data on computer science R&D and overall R&D and used this information to construct variables measuring the sum of computer science and other R&D over the previous five years. The main empirical analyses relate a university's software patenting to its nonsoftware patenting in a given issue year and to its computer

66. In our quantitative discussion, we cannot distinguish between motivations of university technology transfer offices and motivations of university scientists. In any event, these motivations may be related. For example, Azoulay et al., *supra* note 11, at 600, 615–19, find that in a study of 3,862 life sciences researchers, the overall "patent stock" of the university where the researcher is employed has an effect on the number of patents held by the life sciences researcher.

67. In the life sciences context, for example, one study that examined patenting activity for 3,862 researchers found that having co-authors who patent has a positive effect on patenting behavior. *Id.* at 615.

68. This database is available at WebCASPAR: Integrated Science and Engineering Resources Data System, http://caspar.nsf.gov (follow "Info" hyperlink for "NSF Survey of R&D Expenditures at Universities and Colleges") (last visited Apr. 21, 2009).

69. This database uses information from NSF's *Survey of R&D Expenditures at Universities and Colleges*, a survey conducted annually by NSF's Division of Science Resource Statistics. For more on this data source, see *id.* (follow "Info" hyperlink for "NSF Survey of R&D Expenditures at Universities and Colleges").

science R&D and other R&D over the five years prior. We decided to use a five-year time frame because it is difficult to estimate precisely when the research that led to a particular invention disclosure (let alone patent application or issued patent) was conducted. By using a relatively long time frame, we aimed to get a sense of the research commitment of the university, both inside and outside the software arena, around the time an invention disclosure was likely to have been submitted.[70]

To facilitate interpretation, we took natural logarithms of the independent variables.[71] Because the dependent variables are integer valued, we estimated negative binomial regressions relating software patents and pure software patents to research expenditures.[72] Tables 2 through 6 show the main results from these simple cross-sectional regressions for 1982 to 2002, respectively.

Table 2: Negative Binomial Models, Determinants of 1982 Software Patenting

	Model1	Model2	Model3	Model4
	(1)	(2)	(3)	(4)
ln Lagged CS Funds	.218*	.167	.770**	.748**
	(.128)	(.124)	(.336)	(.358)
ln Lagged Other Funds	.612**	-.043	1.435**	.709
	(.270)	(.193)	(.635)	(.774)
ln Other Patents		1.326***		.742
		(.272)		(.501)
Const.	-10.317***	-4.045**	-27.910***	-19.811**
	(2.824)	(1.798)	(8.697)	(10.044)
Obs.	202	202	202	202

Robust standard errors in parentheses. *** denotes p<.001, ** denotes p<.01, and * denotes p<.05.

70. For purposes of our regression analyses, the strong persistence in university-level R&D funding and composition over time is very helpful. As one indicator of this, computer science R&D funding at a point in time has a correlation of 0.965 with its sum over the previous five years, and other R&D funding has a correlation of 0.996 with its sum over the previous five years.

71. Following convention, we use natural logs of one plus the variable.

72. We could not reject the hypothesis of overdispersion and thus chose negative binomial models over Poisson models. However, we obtained qualitatively similar results from Poisson models with standard errors adjusted to account for overdispersion and from log-log models estimated via ordinary least squares. These results are available from the authors on request.

The dependent variable in Models 1 and 2 is the number of software patents issued to a university in a given year. In Models 3 and 4, the dependent variable is the number of pure software patents.

In negative binomial models, coefficients on log-transformed variables can be interpreted as elasticities. Model 1 for the 1982 sample (Table 2) shows that both computer science R&D and other R&D are related to software patenting. Upon introducing the nonsoftware patenting variable, neither of the R&D variables is significant, but the number of nonsoftware patents is, with a one percent increase in other patenting associated with a 1.3% increase in software patenting. Both software and nonsoftware R&D are related to the number of pure software patents (Model 3), but the latter is no longer statistically significant after introducing the nonsoftware patenting variable. Thus, the only clear implication of the 1982 cohort is the strong correlation of nonsoftware patenting with overall software patents.

Table 3: Negative Binomial Models, Determinants of 1987 Software Patenting

	Model1	Model2	Model3	Model4
	(1)	(2)	(3)	(4)
ln Lagged CS Funds	.343***	.314***	.289	.210
	(.122)	(.092)	(.194)	(.162)
ln Lagged Other Funds	1.013***	.291	.864**	.094
	(.236)	(.198)	(.351)	(.319)
ln Other Patents		.797***		1.035***
		(.163)		(.303)
Const.	-16.411***	-8.430***	-15.243***	-6.796**
	(2.592)	(2.117)	(3.792)	(3.261)
Obs.	202	202	202	202

Robust standard errors in parentheses. *** denotes p<.001, ** denotes p<.01, and * denotes p<.05.

Table 4: Negative Binomial Models, Determinants of 1992 Software Patenting

	Model1	Model2	Model3	Model4
	(1)	(2)	(3)	(4)
In Lagged CS Funds	.143*	.082	.122	.071
	(.077)	(.073)	(.110)	(.107)
In Lagged Other Funds	1.083***	.348*	.968***	.264
	(.192)	(.206)	(.287)	(.323)
In Other Patents		1.027***		.955***
		(.199)		(.318)
Const.	-15.295***	-7.610***	-14.735***	-7.392**
	(2.213)	(2.221)	(3.357)	(3.446)
Obs.	202	202	202	202

Robust standard errors in parentheses. *** denotes p<.001, ** denotes p<.01, and * denotes p<.05.

Tables 3 and 4 show that, in 1987 and 1992, once nonsoftware patents are taken into account, neither computer science nor other R&D has a consistently strong association with either overall software patenting or pure software patenting. Nonsoftware patenting is, however, strongly related to software patenting and pure software patenting, after controlling for R&D. Depending on the specific model and gear, elasticities range from 0.797 to 1.035.

Table 5: Negative Binomial Models, Determinants of 1997 Software Patenting

	Model1	Model2	Model3	Model4
	(1)	(2)	(3)	(4)
In Lagged CS Funds	.245***	.172**	.348***	.275**
	(.076)	(.072)	(.134)	(.131)
In Lagged Other Funds	.986***	.308	1.005***	.271
	(.160)	(.196)	(.252)	(.320)
In Other Patents		.832***		.813***
		(.185)		(.282)
Const.	-14.941***	-7.394***	-17.358***	-9.023**
	(1.878)	(2.179)	(2.975)	(3.587)
Obs.	202	202	202	202

Robust standard errors in parentheses. *** denotes p<.001, ** denotes p<.01, and * denotes p<.05.

Table 6: Negative Binomial Models, Determinants of 2002 Software Patenting

	Model1	Model2	Model3	Model4
	(1)	(2)	(3)	(4)
ln Lagged CS Funds	.110**	.097**	.588***	.523***
	(.049)	(.045)	(.121)	(.110)
ln Lagged Other Funds	.839***	.073	.223	-.395**
	(.118)	(.141)	(.161)	(.163)
ln Other Patents		.910***		.781***
		(.141)		(.167)
Const.	-11.577***	-3.613**	-9.017***	-2.182
	(1.406)	(1.510)	(1.615)	(1.576)
Obs.	202	202	202	202

Robust standard errors in parentheses. *** denotes p<.001, ** denotes p<.01, and * denotes p<.05.

Further, Tables 5 and 6 show that, after the mid-1990s—the post-*Alappat*[73] era—universities that had larger stocks of computer science R&D also tended to do more software patenting and pure software patenting. This conclusion is consistent with the descriptive data for issue year 2002 in Table 1. In each of the models, however, after controlling for computer science R&D and other R&D, nonsoftware patenting was significantly related to both software patenting and pure software patenting, with elasticities ranging from 0.78 to 0.91 depending on the specific model and year.

The finding that nonsoftware patenting (conditional on overall and computer science R&D) has a qualitatively and statistically significant relationship with software patenting suggests that universities with a higher nonsoftware patenting propensity also tend to patent more software.

This correlation between nonsoftware and software patenting propensity could be driven by a number of factors. One potentially omitted variable from the analyses above is university success in getting patent applications issued. If some universities are systematically better at securing patents, this could drive the observed correlation even if the propensity to file patent applications on software is unrelated to the propensity to file applications on other inventions. We explored this question for the 2002 cohort for the

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

subset of 130 universities where AUTM data on "application success" were available.[74]

Table 7: Negative Binomial Models, Determinants of 2002 Software Patenting, AUTM Universities

	Model1	Model2	Model3	Model4
	(1)	(2)	(3)	(4)
In Lagged CS Funds	.100**	.537***	.091	.583***
	(.046)	(.111)	(.057)	(.120)
In Lagged Other Funds	-.040	-.524**	.428**	-.008
	(.169)	(.235)	(.202)	(.254)
In Other Patents	.807***	.717***		
	(.160)	(.219)		
Application Success Rate	-.207	.068		
	(.360)	(.426)		
In TTO FTEs			.503**	.407*
			(.208)	(.247)
Const.	-1.663	-.317	-6.420***	-6.356**
	(1.914)	(2.530)	(2.468)	(3.046)
Obs.	130	130	101	101

Robust standard errors in parentheses. *** denotes p<.001, ** denotes p<.01, and * denotes p<.05.

As Models 1 and 2 in Table 7 show, for all software patents and pure software patents respectively, application success is not significantly related to software patenting.

Another possible explanation for the correlation between nonsoftware and software patenting propensity could be scale economies: the marginal costs of obtaining a software patent (or any additional patent) may be lower for universities that have larger technology transfer operations. To test this possibility, we used data on the size of technology transfer offices. Reports from the Association of University Technology Managers contain data on the number of licensing officers employed by each university in 2002, though only for a subset (102) of universities in our sample.[75] Models

74. More specifically, data on the number of patent applications filed by the university were available. Combining this data with PTO data on issued patents based on 2002 applications (as of December 2007) allowed us to measure the university's "application success." Because the application success variable captures university patent prosecution ability based on applications filed in 2002 (and issued over the next five years) rather than in the years prior to the issuance of the 2002 patents, the application success measure is potentially a noisy one. However, we suspect that university success in obtaining patents is relatively invariant over time.

75. Association of University Technology Managers, FY 2002 AUTM Licensing Survey (on file with author).

3 and 4 in Table 7 show that, after controlling for nonsoftware and software R&D, this variable is significantly related to both pure and software patenting, consistent with the scale economies hypothesis.

Finally, we estimated a panel version of these regressions for the entire period with university and year fixed effects. In this model, the estimated changes in R&D and other patenting are identified using within-university variation over time. Thus, any time invariant university factors (including "fixed" university technology transfer abilities) would not affect estimates from this model.

Table 8: Negative Binomial Models, Determinants of Software Patenting, Panel Model

	Model1	Model2
	(1)	(2)
Log Lagged CS Funds	.039	.107
	(.047)	(.096)
Log Lagged Other Funds	-.387	-.118
	(.289)	(.628)
Log Other Patents	.457***	.054
	(.093)	(.167)
Year Dummy 1987	.688***	1.363**
	(.244)	(.583)
Year Dummy 1992	1.381***	2.193***
	(.324)	(.754)
Year Dummy 1997	1.767***	2.708***
	(.400)	(.897)
Year Dummy 2002	2.090***	3.045***
	(.472)	(1.040)
Const.	-17.388	-15.120
	(5379.716)	(418.959)
Obs.	1010	1010

All models include university fixed effects. The left-out year category is 1982. Robust standard errors in parentheses. *** denotes p<.001, ** denotes p<.01, and * denotes p<.05.

Table 8 shows the results. Notably, the year dummies are positive and their magnitude increases over time, reflecting the overall growth in university patenting over the 1982–2002 period. Model 1 shows that the main factor affecting overall software patenting is changes in nonsoftware patenting, with a one percent increase in nonsoftware patenting implying a 0.46% increase in software patenting. The corresponding finding for pure software patenting is neither qualitatively nor statistically significant. Additionally, none of the R&D measures are statistically significant for either software or pure software. This last set of results should be interpreted with caution, however. Because the panels are short and there is limited within-university variation, it is difficult to draw strong inferences.

Taken together, we read these results as consistent with the impressions from our interviews (discussed below) that university software patenting practices are related to their nonsoftware patenting practices. An ideal test of this hypothesis would control for the number of disclosed software inventions for which patents would indeed facilitate technology transfer. If we had such a measure, and found that nonsoftware patent propensity nonetheless affected the volume of software patenting, this would provide strong support for the argument that university software patenting is less about "technology transfer" and more about "business as usual." Unfortunately, such a measure is unlikely to be obtained.[76]

C. *Departmental Origin*

To explore the sources of software patenting in greater detail, we also identified the departmental origin of the inventors in the top fifteen academic software patentees in 2002. As noted above, these institutions accounted for sixty percent of 2002's overall academic software patenting.[77]

These 241 patents had 544 distinct inventors. Based on web searches, we were able to locate the primary departmental affiliation for 73.5% (400) of these inventors. Seen another way, these 241 patents include 661 unique inventor-patent dyads. For example, if an inventor is included on three patents, she generates three inventor-patent dyads. For 26.5% of these dyads, we could not identify the inventor's department.

Table 9: Departmental Origin of Inventors on Software Patents Issued to the Top Fifteen Recipients of Software Patents in Issue Year 2002

Department	N	Percent	Cumulative Percent
EE/CS	47	9.67	9.67
Electrical Engineering	47	9.67	19.34
Computer Science	42	8.64	27.98
Neuroscience	23	4.73	32.72
Robotics Institute	23	4.73	37.45
Medical Physics	20	4.12	41.56
Radiology	19	3.91	45.47
Lawrence Livermore	15	3.09	48.56
Chemistry	13	2.67	51.23
Mechanical Engineering	12	2.47	53.70

Counts and percentages calculated at the patent-inventor level

76. Another limitation of our data is that we must rely on R&D generally (and computer science R&D in particular) as an imperfect proxy for the volume of software output at a university.

77. *See supra* Part II.A.

Table 9 shows the top ten departments for dyads where the department is known. Slightly fewer than thirty percent of the inventors are from electrical engineering, computer science, and joint EE/CS departments. Moreover, consistent with arguments that software is produced across the university, a number of biomedical departments are also represented, including Neuroscience, Radiology, and Medical Physics. To the extent that software is being treated just like other inventions for purposes of patenting, this may be in part because a significant proportion of it emerges from biomedical departments where patenting is the norm. In this regard, it bears emphasis that, of the 970 software patents in our overall sampling, 333 (more than one-third) represented software with biomedical applications.

Table 10: Departmental Origin of Inventors for Pure Software Patents Issued to the Top Fifteen Recipients of Software Patents in Issue Year 2002

Department	N	Percent	Cumulative Percent
Computer Science	33	23.57	23.57
EE/CS	29	20.71	44.29
Electrical Engineering	12	8.57	52.86
Lincoln Laboratory	6	4.29	57.14
Aeronautics and Astronautics	5	3.57	60.71
Robotics Institute	5	3.57	64.29
Agricultural and Biological Engineering	3	2.14	66.43
Chemistry and Biochemistry	3	2.14	68.57
Mathematics	3	2.14	70.71
Biostatistics and Medical Informatics	2	1.43	72.14

Counts and percentages calculated at the patent-inventor level

Table 10 shows the analogous table for pure software patents. Perhaps not surprisingly, more than half of the patent-inventor dyads on pure software patents emanate from electrical engineering, computer science or joint EE/CS departments. Other departments are less prominent in production of pure software patents than they are in the production of other software patents.

III. UNIVERSITY OWNERSHIP POLICIES

Our quantitative results demonstrate that, on the whole, for software patent applications filed in the 1980s and 1990s, universities appeared to take a "one size fits all" approach to technology. Those universities that tended to file a lot of nonsoftware patents also filed a lot of software patents (both ordinary software and pure software).

The regression results underscore what can be seen through a casual review of the descriptive statistics in Table 4: the top five software patentees in 2002—MIT, University of California, Stanford, Caltech, and the University of Texas—were also the top five overall patentees. Notably, none of these five was in the top five in computer science R&D spending for the years 1996–1998.

On the other hand, a qualitative review of our descriptive data also indicates some anomalies in this general pattern. For example, of the top five software patentees in 2002, two—Caltech and the University of Texas—did *not* have significant numbers of pure software patents.[78] Moreover, five universities—the University of Washington, Georgia Tech, Carnegie Mellon University, the University of Rochester, and the University of Illinois—rank substantially higher in software/pure software patenting than in overall patenting.

In order to get a more fine-grained sense of factors that might be influencing individual universities, we investigated technology transfer policies at the fifteen universities that received the most software patents in 2002. In addition to reviewing publicly available materials, we conducted interviews with two groups of relevant parties: (1) technology transfer managers responsible for software at the universities;[79] and (2) academics and graduate students prominent in software development. The latter group was selected through snowball sampling. Interviews were conducted by telephone, lasted approximately forty-five minutes to one hour, and were semi-structured. On occasion, follow-up discussions were necessary and were conducted either via e-mail or by telephone. A list of questions asked in each interview is attached as Appendix A.[80]

As an initial matter, several of the interviews confirmed the results of our quantitative analysis—that technology transfer officers treat software like other inventions. Lita Nelsen, director of the MIT technology transfer office noted that, when discussing pure software, "if there are no strong feelings on the part of the authors to open source their work, we will look at it *like any other invention.*"[81]

78. Caltech and the University of Texas ranked twenty-third and forty-first, respectively.

79. We were able to obtain interviews with officials at all technology transfer offices except Caltech and Columbia University.

80. All interviews were recorded; transcripts of recordings (with portions where interviewees wished to remain anonymous deleted) are available upon request from the authors.

81. E-mail from Lita Nelsen, Director, MIT Tech. Licensing Office, to Colin Crossman, Faculty Fellow, Duke Law Sch. (July 27, 2005, 13:45 EST) (on file with the

Our qualitative investigation also suggested reasons for the relatively low numbers of pure software patents at the University of Texas and Caltech. At the University of Texas, complaints from computer science faculty resulted in the explicit adoption of a policy allowing faculty freedom to share their work as they wanted.[82] Such freedom includes express permission to use the GNU General Public License ("GPL"), a "viral" open source license that requires software source code to be open but also requires those who redistribute the software to make any modifications they have made to the source code available.[83] This policy was adopted in the 1990s and, therefore, would have affected the number of pure software patents that issued in 2002.[84]

The explanation for Caltech's small number of pure software patents is less transparent. Because we were unable to speak with the technology licensing office,[85] our knowledge of the Caltech situation is based on reports from scientists who work there. According to one prominent open source software developer at Caltech, most software developed there is for internal, in-house use and is probably not even reported to the technology transfer office.[86] For software that is reported, the faculty researcher decides how the software should be exploited.[87] The technology transfer office is not only familiar with open source but is apparently eager to use the viral GPL license; the viral GPL allows for the future possibility of "forking," with one fork continuing to be available without charge via the GPL and the other

North Carolina Law Review). According to Nelsen, software tends to be worth patenting when the primary value is in the algorithm and when it is likely to give "a substantial return." *Id.*; *see also* Telephone Interview with Gerald Barnett, Dir., Univ. of Wash., Software & Copyright Ventures (Oct. 2, 2003) (emphasis added) (noting that universities that have a long history of patenting tend to see software, including pure software, through the lens they use for other invention).

82. Telephone Interview with Georgia Harper, Attorney, Intellectual Prop. Section, Univ. of Tex. Office of Gen. Counsel (Feb. 26, 2004).

83. For information about the GPL license, see GNU General Public License, Free Software Foundation, Inc. (3d version, June 29, 2007), http://www.gnu.org/licenses/gpl.html.

84. *See* Telephone Interview with Georgia Harper, *supra* note 82. Various University of Texas websites document the University's continuing support of open source under GPL or similar licenses where the faculty inventor and a software consultant determine that such distribution is in the best interest of the University and the public. *See, e.g.*, Administrative Policy Regarding Disclosure, Distribution and Licensing Software, University of Texas (Dec. 15, 2004), http://www.utsystem.edu/ogc/intellectualproperty/swadmpol.htm.

85. The Caltech technology transfer staff refused requests for an interview.

86. Telephone Interview with C. Titus Brown, Graduate Student, Cal. Inst. of Tech., Dept. of Biology (Aug. 17, 2005).

87. *Id.*

fork converted into a nonviral license for which corporate clients might be willing to pay.[88]

In contrast, the positions of two of the other top five software patentees—MIT and the University of California—are (or at least have been) less explicitly favorable to open source.[89] According to MIT's director of technology transfer, MIT allows researchers to use the open source approach, and even manages their licenses, but this position is not part of an official policy.[90] The University of California system appears to have a highly complex set of positions on open source licensing. While some campuses, such as Berkeley and San Diego, are familiar with the open source model, other campuses are less so.[91] Complexity is compounded by the fact that various important campuses, including Berkeley, have been actively fine-tuning their policies.[92] As of early 2002, officials from the Berkeley office were quoted as criticizing the decision made by Berkeley a decade earlier, in 1992, to release as open source the Unix operating system and TCP/IP networking protocol.[93] At about the same time, Berkeley computational biologist Steven Brenner encountered difficulties in releasing his lab's software under an open source license.[94]

As for the five schools that rank substantially higher in software/pure software patenting than in other patenting—the University of Washington, Georgia Tech, Carnegie Mellon, the University of Rochester, and the University of Illinois—the first three of these schools have unique characteristics that may explain their high levels of pure software patenting. At the University of Washington, a separate technology licensing office, now called the Digital Ventures Office, is responsible for managing pure software

88. *See id.*

89. Where Stanford University fits in the picture is not clear. Stanford did not adopt an explicit policy in favor of open source until 2004. It adopted this policy in response to a number of requests received from professors. *See* Telephone Interview with Katherine Ku, Dir., Stanford Office of Tech. Licensing (Sept. 6, 2005). However, Stanford has long had a policy allowing professors to put their inventions into the public domain if they so desire.

90. Telephone Interview with Lita Nelsen, Dir., Mass. Inst. of Tech. Licensing Office (Aug. 8, 2005).

91. Telephone Interview with William Decker, Assoc. Dir., Univ. of Cal. at San Diego Tech. Transfer Office (Aug. 28, 2003).

92. *See infra* notes 180–83 and accompanying text.

93. *See* Jeffrey Benner, *Public Money, Private Code*, SALON, Jan. 4, 2002, http://www.salon.com/tech/feature/2002/01/04/university_open_source/index.html.

94. Telephone Interview with Steven Brenner, Assoc. Professor, Univ. of Cal. at Berkeley Dept. of Plant & Microbial Biology (Mar. 8, 2004).

(and digital products more generally).[95] By 2005, the office employed seven full-time professionals as well as two to four part-time students.[96] As for Georgia Tech, its level of computer science R&D funding relative to other funding is quite high—it ranks fifth in computer science funding and only thirty-second in other funding. Finally, Carnegie Mellon ranks second in computer science funding and eighty-seventh in other funding. Carnegie Mellon is also home to the Software Engineering Institute ("SEI"), a research consortium founded by software firms.[97] Patents from SEI are assigned to Carnegie Mellon, and the software firms in the consortium receive an automatic, royalty-free license.[98]

IV. UNIVERSITY SOFTWARE OWNERSHIP: A POLICY ANALYSIS

A. General Considerations

Our results indicate that universities have become active patentees of software, including pure software. They have clearly availed themselves of the opportunities afforded by the Bayh-Dole Act and the Federal Circuit decisions in the 1990s. Moreover, at least for our sample—which, because of prosecution lag time, terminates with patents filed around 2000—behavior with respect to total software patents and pure software patents is strongly affected by nonsoftware patent propensity. Although there are anomalies in the overall pattern—largely explicable through policies at particular universities that either strongly favor open source or favor a particular focus on software[99]—the correlation is clear.

From a private point of view, this correlation may make sense, especially if the reason for the effect is, as some of our data suggests, economies of scale at the technology transfer office.[100] To the extent such scale economies exist, they are likely to lower the private

95. Digital Ventures Home Page, http://depts.washington.edu/ventures/ (last visited Apr. 29, 2007).

96. Telephone Interviews with Charles Williams, Dir., Univ. of Wash. Digital Ventures (Apr. 21, 2004 & Sept. 1, 2005).

97. *See* Carnegie Mellon University, Software Engineering Institute, http://www.sei.cmu.edu/ (last visited Apr. 21, 2009).

98. Telephone Interview with Carl P. Mahler II, Dir. of Intellectual Prop., Ctr. for Tech. Transfer & Entm't (Sept. 2, 2005).

99. *See supra* Part III.

100. *See* David C. Mowery & Bhaven N. Sampat, *Patenting and Licensing University Inventions: Lessons from the History of the Research Corporation*, 10 INDUS. & CORP. CHANGE 317, 322–24 (2001).

marginal cost of patent acquisition.[101] From a social point of view, however, patenting based on scale economies is quite problematic. Ideally, we would want decisions about whether to patent publicly funded academic research to be based not only on the private marginal costs of patent acquisition, but also on whether a patent is needed to facilitate commercialization in a specific case, which is likely to vary across inventions and fields.

Stated differently, a lack of differentiation between software and other research could be problematic. As the data on the varying importance of patents as incentives in different industries suggests,[102] and as case study research on the disparate development trajectories of university technologies specifically emphasizes,[103] the optimal mode of university-industry technology transfer is likely to vary by industry and invention.[104] More specifically, as indicated earlier, the theory espoused in the legislative history of Bayh-Dole—that patents and exclusive licenses are necessary to create incentives for firms to develop and commercialize "embryonic" university inventions[105]—does not apply neatly in the software context, where development costs are often low relative to other types of inventions.[106]

Another major argument often advanced in favor of patents is that the prospect of licensing royalties induces university researchers to work with industry licensees and thereby transfer tacit knowledge

---

101. *See id.* at 325.

102. *See* Cohen et al., *supra* note 13, at 5–30 (examining the diverse methods used by R&D labs to protect profits made from their inventions, as well as the diverse motives to patent across different industries).

103. *See generally* Jeannette Colyvas et al., *How Do University Inventions Get into Practice*, 48 MGMT. SCI. 61 (2002) (discussing eleven case studies from Columbia University and Stanford University).

104. In addition to differential incentive effects, university patents and licenses have different informational effects across different industries. Of particular relevance to our study, the comprehensive Carnegie Mellon survey, conducted on a broad range of large and small firms in the early 1990s, indicates that outside of the pharmaceutical and biotechnology industries, industrial R&D managers rate patents and licenses very low relative to other sources of information on public research (e.g., publications, conferences, informal interaction with university researchers, and consulting). *See* Cohen et al., *supra* note 13, at 4–5 (surveying various sizes of firms and their R&D managers). Even within the pharmaceutical industry, patents and licenses were less important than research publications and conferences. *Id.*

105. *See supra* note 12 and accompanying text.

106. To be sure, such costs may be higher in situations where the software is not pure software. However, even in those cases, to the extent that the novel or nonobvious element is likely software, development costs are probably still low relative to the biopharmaceutical industry.

necessary for commercialization.[107] Although this argument could in theory be compatible with exclusive or nonexclusive licenses, the assumption tends to be that an academic researcher would have sufficient time for only one exclusive licensee. However, in comparison to the life sciences, software (particularly pure software) is an area of invention where knowledge is likely to be relatively codified. Object-oriented programming is based on principles of modular design, and one of the reasons that open source methods of software production have been successful is that the development task can be broken up into modular pieces that are then reassembled.[108] So, the need for transfer of tacit knowledge may not be as pervasive as it is in the life sciences.

Indeed, in some well known cases, it appears that the university patent allowed the university and/or its exclusive licensee to extract rents from other firms without aiding in technology transfer. For example, in the prominent case of *Eolas Technologies, Inc. v. Microsoft Corp.*,[109] it does not appear that Microsoft's commercialization was aided by the activities of the University of California/Eolas.[110] Even worse, in *Eolas*, the university patent was arguably being used to "holdup" the commercializer—that is, to use the threat of injunctive relief to extract rents significantly larger than the inventive contribution made by the patentee (and thus also significantly larger than any licensing fee the university inventor could have charged *ex ante*).[111]

---

107. The evidence generally cited for this argument is survey data presented in Richard Jensen & Marie Thursby, *Proofs and Prototypes for Sale: The Licensing of University Inventions*, 91 AM. ECON. REV. 240, 240 (2001). The Jensen and Thursby survey of sixty-two technology transfer offices ("TTO") found that TTO managers thought that inventor involvement was often important in the commercialization of inventions. *Id.* at 255.

108. *See* Yochai Benkler, *Coase's Penguin, or, Linux and* The Nature of the Firm, 112 YALE L.J., 369, 383–88 (2002).

109. 399 F.3d 1325 (Fed. Cir. 2005).

110. According to a University of California website on the *Eolas* case, Microsoft and other firms were approached about licensing the technology in 1994 but declined to do so. University of California, Questions and Answers About UC/Eolas Patent Infringement Suit Against Microsoft, http://www.ucop.edu/news/archives/2003/aug11art1qanda.htm (last visited Apr. 21, 2009).

111. The legal and economic literature on holdup through property rights assertion (both within and outside the patent context) is voluminous. *See, e.g.*, Arti K. Rai, *Regulating Scientific Research: Intellectual Property Rights and the Norms of Science*, 94 NW. U. L. REV. 77, 127–28 (1999) (summarizing literature). For a recent contribution, see Mark Lemley & Carl Shapiro, *Patent Holdup and Royalty Stacking*, 85 TEX. L. REV. 1991, 2008–10 (2007).

B. *The Problem of Holdup Litigation*

In order to study the litigation question more systematically, we conducted two somewhat related empirical inquiries.[112] First, we used four different search methods to collect case studies involving university software litigation.[113] Second, by running the patents in our database against the Derwent LitAlert database of patent cases filed, we determined whether the software patents in our sample were litigated at rates higher (or lower) than the nonsoftware patents in our sample.[114]

With respect to the latter inquiry, there was no statistically significant difference in rates of litigation between the software and nonsoftware patents in our sample.[115] However, we did generate a number of case studies in which university software patents appear to have been used in a manner that hindered, rather than promoted, commercialization. Research Corporation Technologies ("RCT"), a firm that is the assignee of patents from the University of Rochester, has actively pursued litigation on a group of six patents covering the

112. To our knowledge, empirical work on university patent litigation has not been technology specific. The empirical work to date indicates that, across all technologies, university patents are litigated at about the same rate as other patents; that such litigation has tripled in the period from 1985–2000; and that litigation tends to consume time and resources that might otherwise be spent in marketing technology and establishing licenses. *See* Scott Shane & Deepak Somaya, *The Effects of Patent Litigation on University Licensing Efforts*, 63 J. ECON. & BEHAV. ORG. 739, 740–41 (2007).

113. Specifically, we gleaned our cases from four sources: (1) searches of the U.S. Patent Quarterly ("USPQ") database for cases where a university was a party to the litigation; (2) searches of the Derwent LitAlert database for cases where a university was a litigation party; (3) searches of general news databases for discussions of university-related patent litigation; and (4) running the patent numbers for all university software patents in our database against the LitAlert database. All inquiries were conducted through September 2006. When we found cases of interest, we used PACER docket records to supplement our inquiry. In general, our discussion of cases is unlikely to be biased based on underinclusiveness. The USPQ database is underinclusive to the extent that it contains information only about cases from which a written order has emerged. Although the LitAlert database is supposed to include all cases involving a patent, it actually includes a little over half of such cases. *See* Deepak Somaya, *Strategic Determinants of Decisions Not to Settle Patent Litigation*, 24 STRATEGIC MGMT. J. 17, 21–22 (2008). However, there is no reason to believe the underinclusiveness is biased in favor of, or against, software patent litigation.

114. This patent search includes all cases filed before September 2006. As noted earlier, although this database is underinclusive, there is no reason to believe its underinclusiveness is biased in favor of or against software patent litigation. These results are available upon request.

115. In part, this may be because the total numbers (five software cases and forty-six nonsoftware cases filed with respect to the patents in our sample) were small.

so-called Blue Noise Mask printing technology.[116] This technology, which was developed at the University of Rochester and assigned to RCT under a technology evaluation and commercialization agreement dating back to 1953, allows for high-quality "half-tone" printing.[117] It is used by firms ranging from Hewlett-Packard to Microsoft.[118] The firm's website emphasizes the patent suits it has brought against Hewlett-Packard, Epson, and Microsoft and "invites current and potential users of this landmark technology to contact [RCT] about licenses under RCT's patent rights."[119] The lawsuits against Hewlett-Packard and Epson were settled in 1999 and 2001, respectively.[120] The litigation against Microsoft is ongoing.[121]

MIT and its exclusive licensee, Akamai, were recently involved in lawsuits against two firms, Speedera Networks Inc. and C&W Wireless Internet Services, that allegedly infringed the MIT/Akamai patent on software for decreasing congestion and delay in accessing web pages on the Internet.[122] The Akamai technology, which was launched in 1999 at the height of the "dot-com" boom, was similar to technologies developed by other firms before Akamai.[123] The district court granted a permanent injunction upon finding that the patent was valid and had been infringed.[124] On appeal, however, the Court of Appeals for the Federal Circuit determined that the broadest claims of the patent were invalid, because the C&W software actually predated them.[125]

In other cases, university patents have been asserted for large damage awards (usually for the statutory maximum of six years of past infringement) immediately before they are about to expire. For example, MIT and an exclusive licensee, Electronics for Imaging, have sued ninety-two firms, including Microsoft and IBM, alleging

116. *See* Research Corporation Technologies, Licensing: Blue Noise Mask, http://www.rctech.com/licensing/lic-blue-noise-mask.php (last visited Apr. 21, 2009).

117. *See id.*

118. *See id.*

119. *See id.*

120. *See id.*

121. Research Corp. Techs. v. Microsoft Corp., No. CV-01-658-TUC-MLR, 2006 U.S. Dist. LEXIS 66875 (D. Ariz. June 13, 2006), *vacated*, 536 F.3d 1246, 1247 (Fed. Cir. 2008).

122. *See* Akamai Techs., Inc. v. C&W Internet Servs., 344 F.3d 1186, 1188–89 (Fed. Cir. 2003).

123. *Id.* at 1193 (quoting *Akamai* brief noting this point).

124. *Id.* at 1188.

125. *Id.* at 1194–95. Even the narrower claims were found valid only because the Federal Circuit required C&W to establish that there be a "suggestion or motivation to combine" prior art references. *Id.* at 1196. This "suggestion" test has been controversial and the rigid application of it was recently rejected by the Supreme Court in *KSR v. Teleflex*, 550 U.S. 398, 418 (2007).

infringement of a patent covering image-editing software.[126] This lawsuit was filed in December 2001, six months before the patent was set to expire.[127] The technology in question is a color imaging method that can be applied to any system that produces color pictures.[128] Similarly, in March and May of 2005, a few months before the relevant patent was due to expire, the University of Texas filed three lawsuits against a total of forty-two electronics manufacturers alleging infringement on a software patent that allows text messaging through a standard telephone keypad.[129] In some cases, the patent in question may not be due to expire immediately, but it is nonetheless relatively old. In 2002, for example, Cornell sued Hewlett-Packard over a 1989 patent obtained by a Cornell professor for a technique that accelerates a computer's processing speed.[130] Similarly, a 2001 suit by MIT against Lockheed-Martin involved a 1989 patent on systems for analyzing acoustic waveforms.[131]

In all of these cases, commercialization by firms other than the university licensee was going forward, and patent rights/exclusive licenses do not appear to have been necessary to facilitate "technology transfer."[132] Moreover, there is no evidence in these cases that the other firms' development efforts were "free-riding" on licensees' investments.[133] In fact, in at least three of these cases, the patent had not (apparently) been licensed at all.[134] Contrary to the spirit of Bayh-Dole, these patents simply allowed universities to extract rents and perhaps even holdup development efforts.[135]

---

126. Mass. Inst. of Tech. v. Abacus Software, 462 F.3d 1344, 1349 (Fed. Cir. 2006).

127. *Id.*

128. In April 2002, MIT and EFI expanded their complaint to include 214 defendants. In the course of litigation, plaintiffs settled with some defendants and dismissed their claims against others, so that only four remained: Corel, Microsoft, Roxio, and MGI Software. *Id.* at 1344. As discussed further below, the Federal Circuit recently overturned a district court claim construction that favored the defendants in this case. *See infra* note 142 and accompanying text.

129. Bd. of Regents v. Benq Am. Corp., 533 F.3d 1362, 1365 (Fed. Cir. 2008).

130. *See* Cornell Univ. v. Hewlett-Packard Co., 313 F. Supp. 2d 114, 120 (N.D.N.Y. 2004).

131. Mass. Inst. of Tech. v. Lockheed Martin Global Telecomms., Inc., 251 F. Supp. 2d 1006, 1007 (D. Mass. 2003) (granting defendant's motion for summary judgment on noninfringement).

132. *See supra* notes 116–31 and accompanying text.

133. *See supra* notes 116–31 and accompanying text.

134. *See, e.g.*, *Benq America Corp.*, 533 F.3d at 1365 (listing only the University of Texas as a plaintiff); *Cornell Univ.*, 313 F. Supp. 2d at 118 (naming only Cornell University and its research foundation as plaintiffs); *Lockheed Martin*, 251 F. Supp. 2d at 1007 (naming only MIT as a plaintiff).

135. *See supra* note 12 and accompanying text.

Attempts to extract rents from firms that have commercialized successfully may be a particular concern where the case is ultimately very weak. Thus, it is notable that in a number of litigated cases, the university's argument has been unequivocally rejected.[136] For example, the University of Texas recently lost a case involving patents on positron emission technology ("PET"), a medical imaging technology that uses gamma rays to detect cancer, heart disease, brain disorders, and other health conditions.[137] The University of Texas claimed that CTI Molecular Imaging Inc., a leading provider of PET equipment, infringed two of its patents.[138] In that case, both the district court and the Federal Circuit held that the defendant did not infringe.[139] Similarly, in a recent suit brought by the University of California and its exclusive licensee over patented software for eliminating edge artifacts when compressing digital images, both the district court and the Federal Circuit found that the patent in question was neither invalid nor infringed.[140] In 2003, a lawsuit by the University of Illinois against Fujitsu on software patents relating to plasma display panels resulted in a summary judgment determination that the relevant patent claims were invalid.[141] In contrast, our search found only one case in which a university's assertions regarding its patented software were largely vindicated by the court system, either in a final district court decision that was not appealed or in an appellate court decision.[142]

Even if patenting and exclusive licensing of software do not facilitate commercialization *per se*, one might argue that exclusive licenses to university patents are useful for software start-ups, and promoting such start-ups is a socially valuable goal.[143] But, even assuming that the "generating start-ups" argument has merit, the force with which it applies to software is not clear. While most small

136. In the cases discussed above, by contrast, the university patentee lost on many of its arguments but won on some others.

137. Bd. of Regents v. CTI Molecular Imaging, Inc., 164 F. App'x 982, 982 (Fed. Cir. 2005) (per curiam) (unpublished); *see CTI Molecular Imaging, Inc. Announces Dismissal of Lawsuit*, BIOSPACE, Feb. 28, 2005, http://www.biospace.com/news_story.aspx?NewsEntityId=19216020.

138. *See CTI Molecular Imaging, Inc.*, 164 F. App'x at 982.

139. *See id.*

140. LizardTech. Inc. v. Earth Res. Mapping Inc., 424 F.3d 1336, 1337 (Fed. Cir. 2005).

141. Competitive Techs. v. Fujitsu Ltd., 286 F. Supp. 2d 1161, 1209 (N.D. Cal. 2003).

142. Mass. Inst. of Tech. v. Abacus Software, 462 F.3d 1344, 1347 (Fed. Cir. 2006).

143. *See, e.g.*, Ashish Arora & Robert P. Merges, *Specialized Supply Firms, Property Rights, and Firm Boundaries*, 13 INDUS. & CORP. CHANGE 451, 451 (2004). Although Arora and Merges do not specifically discuss university-generated research, the logic of their argument presumably applies to such research.

biotechnology firms that receive venture financing have patents, the available empirical evidence indicates that most software start-ups that receive venture financing, particularly in the first round, do not have patents.[144] Moreover, though there is some evidence that small software firms with patents tend to fare better than firms without such patents,[145] the evidence also indicates that the disparity is much less significant than the disparity between biotechnology firms with and without patents.[146] At a minimum, the "generating start-ups" argument for patenting software is less compelling than it is for biotechnology. In any event, in many of the cases we found, the university was suing on its own behalf; it did not appear to have found a licensee.[147]

Of course, litigated cases are not likely to be representative of all university software patents. There are certainly cases of successful commercialization where the firm in question did have an exclusive license to a university patent. The company Google, which was the exclusive licensee of a patent assigned to Stanford,[148] is a prominent example, as is RSA Security, the exclusive licensee of various data-encryption patents assigned to MIT.[149] Moreover, as noted earlier, university software patents do not appear to be litigated at rates significantly different from those of university patents in other fields.[150] Thus, we cannot state unequivocally that the incidence of holdup in the software patent context is higher than that for university patents in other fields.

---

144. *See* Ronald J. Mann & Thomas W. Sager, *Patents, Venture Capital, and Software Start-ups*, 36 RES. POL'Y 193, 197 (2007). One limitation of the Mann and Sager data, however, is that it includes only the patents that software firms hold in their own names. *Id.* It does not include patents to which firms may have exclusive licenses. *Id.* Another, perhaps more important, limitation is that the Mann and Sager study firms received venture financing during the 1997–1999 "bubble" years, before the 2001 market crash. *Id.* at 195.

145. *See id.* at 194–204; *see also* Cockburn & MacGarvie, *supra* note 49, at 33–34 (finding that in software markets where patenting is intense, entry for firms with patents is easier than entry for firms without patents).

146. Mann & Sager, *supra* note 144, at 194–204.

147. *See, e.g.*, Cornell Univ. v. Hewlett-Packard Co., 313 F. Supp. 2d 114, 118 (N.D.N.Y. 2004); Mass. Inst. of Tech. v. Lockheed Martin Global Telecomms., Inc., 251 F. Supp. 2d 1006, 1007 (D. Mass. 2003).

148. Exclusive Patent License Agreement Dated July 2, 2001 Between Google and Stanford, http://www.techagreements.com/agreement-preview.aspx?num=66564 (last visited Apr. 21, 2009).

149. *See* RSA Security Inc. Company History, http://www.fundinguniverse.com/company-histories/RSA-Security-Inc-Company-History.html (last visited Apr. 21, 2009). Whether patents and exclusive licenses were particularly important to commercial success in these cases is unclear, however.

150. *See supra* note 115 and accompanying text.

Nonetheless, there is reason to believe that holdup is more likely in software than in other fields, mainly because patents and exclusive licenses are less likely to be important for commercialization in this field than in others. To put the point another way, the ratio of false positives (patenting and giving an exclusive license when it is not necessary for commercialization) to false negatives (failing to patent and give an exclusive license when it is necessary for commercialization) is likely to be higher in software than in the life sciences. In this regard, it bears mention that some of the most successful cases of software commercialization have not involved patents and exclusive licenses. For example, a number of unpatented Stanford programs have been widely adopted by the industry: both MINOS, a linear and nonlinear optimization program, and Genscan, a gene structure prediction program, have been used (via a copyright license) by dozens of different commercial firms.[151]

V. THE WAY FORWARD

As the Stanford example suggests, models of successful software commercialization absent patents are possible. More generally, the future of university software ownership may not be one in which universities are using patents as holdup opportunities.

A. *The Impact of* eBay v. MercExchange *and* In re Bilski

As an initial matter, the holdup threat that patentees can wield may have been mitigated by the Supreme Court's decision in *eBay Inc. v. MercExchange, L.L.C.*[152] In that case, the Court made it clear that permanent injunctive relief is not automatic upon a finding of patent infringement.[153] Rather, the plaintiff must prove each element of the traditional four-factor test for permanent injunctive relief: (1) that it has suffered irreparable harm; (2) that remedies available at law, such as monetary damages, are inadequate to compensate for that harm; (3) that the balance of hardship between the patentee and infringer argues in favor of a remedy in equity; and (4) that the public interest would not be disserved by a permanent injunction.[154] Although the Supreme Court opinion explicitly addresses only permanent injunctive relief,[155] it has been interpreted by the lower

151. *See* Katherine Ku, *Software Licensing: Stanford's Approach*, COMPUTING RES. NEWS, Jan. 2002, at 3, 8.

152. 547 U.S. 388 (2006).

153. *Id.* at 391–93.

154. *Id.*

155. *Id.* at 394.

courts as raising the burden faced by plaintiffs in requests for preliminary injunctive relief.[156] The application to preliminary injunctive relief is important because, as was indicated by the famous $612.5 million settlement that RIM, the maker of the BlackBerry device, paid on patents that had been seriously questioned by the PTO illustrates,[157] even the threat of a preliminary injunction—before validity and infringement have been proven—can be used to extract large settlements.

Lower court cases interpreting *eBay*, particularly in the context of permanent injunctive relief, have tended to emphasize that irreparable harm should be presumed, and injunctive relief is thereby available, where the patentee and the infringer are direct competitors; in that case, infringement is likely to cause the patentee to lose market share and profits.[158] In an instructive opinion, the *eBay* district court noted on remand that irreparable harm may also be found where a patentee is seeking to develop its patent in partnership with others (the classic Bayh-Dole rationale).[159] In contrast, where the patentee secures revenues by approaching firms that have already developed, so as "to maximize the value of a license," monetary damages should be sufficient to compensate for infringement.[160] Thus, at least in those situations where the university patent is being asserted without a partner licensee that practices the patent in

156. *See* The Fire of Genius, Injunction (Dec. 31, 2007), http://www.thefireofgenius.com/injunctions/ (noting that, as of December 31, 2007, twelve cases had cited *eBay* in denying requests for preliminary injunction).

157. *See supra* note 10 and accompanying text.

158. For cases emphasizing direct competition when discussing irreparable harm, see generally Smith & Nephew, Inc., v. Synthes, 466 F. Supp. 2d 978, 983 (W.D. Tenn. 2006), *appeal dismissed*, 269 F. App'x 972 (Fed. Cir. 2008); Tivo Inc. v. Echostar Commc'ns Corp., 664 F. Supp. 2d. 664, 669 (E.D. Tex. 2006), *aff'd in part, rev'd in part, and remanded by* 516 F.3d 1290 (Fed. Cir. 2008); z4 Techs. v. Microsoft Corp., 434 F. Supp. 2d 437, 440 (E.D. Tex. 2006), *aff'd*, 507 F.3d. 1340 (Fed. Cir. 2007); Visto Corp. v. Seven Networks Inc., No. 2:03-CV-333-TJW, 2006 WL 3741891, at *4 (E.D. Tex. Dec. 19, 2006); Paice LLC v. Toyota Motor Corp., No. 2:04-CV-211-DF, 2006 WL 2385139, at *4 (E.D. Tex. Aug. 16, 2006), *aff'd in part, rev'd in part, and remanded by* 504 F.3d. 1293 (Fed. Cir. 2007).

159. MercExchange, L.L.C. v. eBay, Inc., 500 F. Supp. 2d 556, 572 (E.D.Va. 2007), *appeal dismissed*, 273 F. App'x 856 (Fed. Cir. 2008) & 273 F. App'x 857 (Fed. Cir. 2008). In this regard, the district court's opinion was in accord with the Supreme Court's refusal to categorically exclude universities from the possibility of injunctive relief. *See* eBay Inc. v. MercExchange, L.L.C., 547 U.S. 388, 393 (2006) ("[S]ome patent holders, such as university researchers or self-made inventors, might reasonably prefer to license their patents, rather than undertake efforts to secure the financing necessary to bring their works to market themselves. Such patent holders may be able to satisfy the traditional four-factor test, and we see no basis for categorically denying them the opportunity to do so.")

160. *MercExchange, L.L.C. v. eBay, Inc.*, 500 F. Supp. 2d at 572.

competition with infringers, one might suppose that the university's post-*eBay* bargaining leverage would be reduced.

The question of how universities should be treated for purposes of injunctive relief was the subject of a recent Federal Circuit appeal, *Commonwealth Scientific and Industrial Research Organisation v. Buffalo Technology Inc.*[161] Although the patentee in that case, Commonwealth Scientific and Industrial Research Organisation ("CSIRO"), is an Australian research institute, not a U.S. university, it generates and licenses technology in precisely the same manner as do U.S. universities.[162] Moreover, the patent in question was a software patent that covers wireless LAN technology, specifically a number of 802.11 standards.[163] In *CSIRO I* and related cases, CSIRO asserted its patent against not only Buffalo Technology, but also Toshiba, SMC, 3Com, and Microsoft.[164] The district court decision granting injunctive relief to CSIRO emphasized the *eBay* Court's reluctance to categorically rule out injunctive relief for universities.[165] The court then went further, ordering injunctive relief on the theory that a failure to grant such relief would create irreparable harm by making it difficult for CSIRO to license its patents and thus support its research enterprise.[166] Therefore, the extent to which university patentees could continue to exercise the threat of holdup would appear to have turned on the Federal Circuit's resolution of *CSIRO I*. However, the Federal Circuit remanded the case on invalidity grounds without ruling on the issue of injunctive relief.[167]

---

161. 542 F.3d 1363 (Fed. Cir. 2008), *reh'g denied*, No. 2007-1449, 2008 U.S. App. LEXIS 25268 (Fed. Cir. 2008) (*CSIRO I*).

162. Commonwealth Scientific & Indus. Research Org. v. Buffalo Tech., Inc. (*CSIRO II*), 492 F. Supp. 2d 600, 604 (E.D. Tex. 2007), *remanded*, No. 2007-1449, slip op. at 36 (Fed. Cir. Sept. 19, 2008), *available at* http://www.cafc.uscourts.gov/opinions/07-1449.pdf.

163. *CSIRO I*, 542 F.3d at 1367.

164. Dell, Intel, and Marvell are plaintiffs in declaratory judgment actions against CSIRO. Microsoft Corp. v. Commonwealth Scientific & Indus. Research Org., 572 F. Supp. 2d 786, 792 (E.D. Tex. 2008) (describing the suits filed against Toshiba, SMC, 3com, and Microsoft). Dell, Intel, and Marvell are plaintiffs in declaratory judgment actions against CSIRO. *See* Microsoft Corp. v. Commonwealth Scientific & Indus. Research Org., Nos. 6:06 CV 549, 6:06 CV 550, 2007 WL 4376104, at *1 (E.D. Tex. Dec. 13, 2007) (describing the declaratory judgment actions brought by Dell, Intel, and Marvell against CSIRO).

165. *CSIRO II*, 492 F. Supp. 2d at 603–04 (E.D. Tex. 2007), *remanded*, No. 2007-1449, slip op. at 36 (Fed. Cir. Sept. 19, 2008), *available at* http://www.cafc.uscourts.gov/opinions/07-1449.pdf.

166. *CSIRO II*, 492 F. Supp. 2d at 604.

167. *See* Commonwealth Scientific & Indus. Research Org. v. Buffalo Tech., Inc., No. 2007-1449, slip op. at 36 (Fed. Cir. Sept. 19, 2008), *available at* http://www.cafc.uscourts.gov/opinions/07-1449.pdf.

As noted earlier, the Federal Circuit recently decided the much-anticipated case of *In re Bilski*.[168] In *Bilski*, the court held that processes (including, presumably, software processes) have to effect a physical transformation or be "tied to a machine" in order to be patentable.[169] Because the *Bilski* decision explicitly leaves open the question of whether recitation of a computer satisfies the "machine" requirement, its impact on the enforcement of existing university software patents is difficult to gauge fully. To the extent that *Bilski* has an impact, however, that impact is likely to be limited to pure software patents. Moreover, if the recitation of a computer is deemed to satisfy the machine requirement, virtually all of the pure software patents in our study should pass muster.

B. Revenue Generation Without Holdup

Contrary to the district court's theory in *CSIRO II*, university interest in revenue generation can be achieved without the innovation-chilling threat of holdup. Specifically, the alternative of ex ante nonexclusive licensing at reasonable rates (which CSIRO allegedly *did not* attempt to do despite a promise to the Institute of Electrical and Electronics Engineers ("IEEE") to license on "reasonable and nondiscriminatory" terms[170]) could be a mechanism for satisfying university interests while minimizing potential harms.[171] Although nonexclusive licenses are something of a tax on commercialization, a small royalty rate is unlikely to deter most commercialization.

Sound practical reasons may, however, counsel against nonexclusive licensing of *patents*. Although, as noted earlier, economies of scale may reduce prosecution costs for large technology transfer offices,[172] the prosecution of patents still requires some investment of money and staff time. While exclusive licensees typically pay patent application costs, nonexclusive licensees

168. 545 F.3d 943, 966 (Fed. Cir. 2008)

169. *Id.*

170. Brief of Accton Technology Corp. (Taiwan) et al. as Amici Curiae in Support of Appellants Buffalo Technology (USA), Inc. and Buffalo, Inc. and Reversal of Injunction Order at 6–7, Commonwealth Scientific & Indus. Research Org. v. Buffalo Tech., Inc., No. 2007-1449 (Fed. Cir. Sept. 19, 2008), *available at* http://www.mmblaw.com/media/news/media.326.pdf (using RAND as an abbreviation for reasonable and nondiscriminatory).

171. *See, e.g.*, Ku, *supra* note 151, at 8 (indicating that Stanford uses nonexclusive licenses to make software available for "modest fees").

172. *See supra* note 100 and accompanying text.

generally do not.[173] Notably, however, software (particularly pure software) differs from virtually every other university invention in that patents do not have to serve as the foundation of a nonexclusive licensing scheme. Copyright, which attaches without cost upon creation of the software, will do the job.[174] For this reason, at least some technology transfer offices state that they are beginning to refrain from seeking patents.[175] At the University of Washington, for example, the Digital Ventures office says that it uses patenting only if there is a real need to improve the technology (presumably via an exclusive license).[176] Digital Ventures has also convinced start-ups to "go without a patent."[177] For example, MINOS, which is available via a nonexclusive copyright license, is one of Stanford's largest money generators.[178]

Because copyright has been interpreted by the courts to cover little more than source code,[179] it generally does not, by itself, confer a great deal of power. For this reason, commentators concerned about the negative effects of strong proprietary rights have typically focused on patents. Even so, firms appear to be willing to license software because they do not want to bear the costs of independent creation.

On the other hand, even nonexclusive copyright licensing with relatively low fees can be problematic for nonprofit researchers. Thus, universities that want to balance the goal of academic access with revenue generation, such as the University of Washington, are assessing which licenses and royalty structures are appropriate for which situations.[180] The technology transfer offices at the University of Washington, the University of California at Berkeley, and Stanford University have all embraced the idea of dual licenses that give relatively inexpensive access to the nonprofit sector but allow for

173. Statement from Joel B. Kirschbaum, Ph.D., UCSF Office of Technology Management, Non-exclusive Patent Licenses—Who Benefits? (on file with the North Carolina Law Review).

174. *See* Ku, *supra* note 151, at 8.

175. *See* Frequently Asked Questions, Office of Intellectual Property and Industry Research Alliances, University of California, Berkley, http://ipira.berkeley.edu/page.php?nav=75 (last visited Apr. 21, 2009).

176. E-mail from Charles Williams, Director, Tech. Transfer, Univ. of Or., to Arti K. Rai, Prof., Duke Univ. Sch. of Law (Sept. 2, 2005 01:19 EST) (on file with the North Carolina Law Review).

177. *Id.*

178. Telephone Interview with Katherine Ku, *supra* note 89.

179. *See supra* note 42 and accompanying text.

180. *See Meet Charles Williams*, INGENUITY, Summer 2004, at 1, 2, *available at* http://depts.washington.edu/techtran/aboutus/Newsletter/Summer04.pdf.

revenue generation from the commercial sector.[181] Indeed, Stanford's lucrative MINOS business software program is available via a dual copyright license—the commercial use costs more than ten times as much as the academic use.[182]

C. Evolving Policies at "Pro-Patent" Institutions?

Our research has also generated some evidence of evolving policies at institutions that have historically sought large numbers of software patents. In 2004, the University of California at Berkeley announced a policy that appeared to give a relatively strong endorsement to the open source model. In this new policy, the Berkeley Technology Transfer Office states that it will work with researchers who want to release their code under an open source model.[183] Although it is not clear whether researchers have the final say in cases of conflict, the new policy appears encouraging towards open source. Berkeley has also developed, and encourages researchers to use, an "academic" license, under which source code is made available free of charge to academic institutions and nonprofits, while being made available for a fee to commercial users.[184] In general, the new Berkeley policy appears to encourage the possibility of releasing software under different types of licenses for different purposes.

Moreover, although the University of Washington, Georgia Tech, and Carnegie Mellon have a significant software patent presence, this level of patenting may not represent a continuing pattern. Current technology transfer officials assert that they promote other models of software ownership. In fiscal year 2005, the University of Washington's Digital Ventures office generated ninety percent of its revenue from copyright licensing.[185] The Digital Ventures web site features a large variety of unpatented software that can be secured for commercial use through an online license with a standard rate.[186] Like Berkeley, the University of Washington also encourages "forking" in its licenses—licenses in which software (and

181. *See, e.g.*, Frequently Asked Questions, *supra* note 175.

182. *See* Stanford Business Software Inc., Minos 5.5, http://www.sbsi-sol-optimize.com/asp/sol_product_minos.htm (last visited Apr. 21, 2009).

183. Frequently Asked Questions, *supra* note 175 (discussing open source options available on Berkeley's software disclosure form).

184. *See id.*

185. E-mail from Charles Williams, *supra* note 176.

186. University of Washington, Digital Ventures: Express Licensing Program, http://depts.washington.edu/ventures/UW_Technology/Express_Licenses/ (last visited Apr. 21, 2009).

underlying source code) is made available free of charge for certain uses and available for a fee for other uses.[187] Officials at Georgia Tech's technology licensing office note that they oversee dozens of open source-type software licenses each year and that, given the short technology life cycle in the software industry, algorithm patents are often not very valuable.[188] Finally, at Carnegie Mellon, the current technology transfer team states that it almost never patents unless another entity is willing to pay for such patenting.[189]

Furthermore, information technology can substantially facilitate the task of low-transaction cost, nonexclusive licensing. Universities such as Stanford already make software like MINOS available through simple web-based interfaces.[190] Several dozen universities are currently participating in the Kauffman Foundation's web-based I-Bridge project, which aims to reduce the transaction costs associated with licensing: many of the projects available on the I-Bridge site represent software.[191] Additionally, some large technology firms have decided to sponsor software projects at universities under an explicit "open collaboration" model.[192] In 2005, four technology firms (IBM, Hewlett-Packard, Intel, and Cisco) announced a set of "Open Collaborative Research" principles under which certain types of sponsored research in software would be made freely available.[193] IBM is now embarking on specific software projects at seven universities under the rubric of these principles.[194]

D. The Role of Open Source

The related question of using copyright to promote open source within the university is an interesting one. As we have noted, some universities have long embraced open source, and these universities tend to have smaller numbers of pure software patents.[195] But even among technology transfer officials sympathetic to the goals of open source, some mention difficulties that open source may create in the

187. *Id.*

188. Telephone Interview with Kevin Wozniak, Interim Dir. of Tech. Licensing, Ga. Tech. Research Corp. (Sept. 1, 2005).

189. Telephone Interview with Carl P. Mahler II, *supra* note 98.

190. *See* Minos 5.5, *supra* note 182.

191. Ibridge, http://www.ibridgenetwork.org/ (search "Search Innovations" for "software") (last visited Apr. 21, 2009) (listing participating organizations).

192. *See* Lohr, *supra* note 17.

193. *Id.*

194. *Id.* These universities are Berkeley, Carnegie Mellon, Georgia Tech, the University of Illinois, Rensselaer, Stanford, and the University of Texas, Austin.

195. *See supra* notes 82–88 and accompanying text.

university setting.[196] For example, faculty may prefer open source as a method of distribution, not because of ideological commitment, but because open source-related consulting revenues, unlike licensing royalties, do not have to be shared with the university.[197] Indeed, at least one prominent technology transfer officer (who preferred to remain anonymous) believes that some faculty make software open source for the purpose of attracting widespread interest, but have every intention of asserting proprietary rights over the source code at some later point.[198] Additionally, software is often developed by groups, and technology transfer officers sometimes find themselves in the middle of disputes among group members about the best open source mechanism to use (or, indeed, whether open source should be used at all).[199] Technology transfer officers are also wary that particular types of open source licenses will conflict with obligations to sponsors, including the federal government under Bayh-Dole.[200] Thus, to the extent funding agencies are interested in an open source approach because they think such an approach is likely to produce better software, they need to be aware of possible institutional impediments.

One final point bears emphasis. Because our quantitative data concludes with patents filed in the late 1990s, we cannot say that university software holdings have continued to grow. Indeed, as noted, technology transfer officers from universities with significant software patent holdings, such as the University of Washington and Carnegie Mellon, state that they are now quite receptive to

196. *See supra* notes 81–88 and accompanying text.

197. E-mail from Patrick Jones, Director, Office of Tech. Transfer, Univ. of Ariz., to Arti K. Rai, Prof., Duke Univ. Sch. of Law (Apr. 3, 2004) (on file with the North Carolina Law Review).

198. This technology transfer officer did not specify precisely how a faculty member would assert proprietary rights. In the context of a viral license, one mechanism for doing so would be to "fork" the license. One prong of the license would remain viral while the other, made available to paying customers, would not be viral in character. The software producer MySQL has adopted this strategy. *See* MySQL, Commercial License for OEMS, ISVs, and VARs (Oct. 9, 2008), http://www.mysql.com/about/legal/licensing/oem/.

199. Telephone Interview with Lita L. Nelsen, *supra* note 90; Telephone Interviews with Charles Williams, *supra* note 96.

200. In theory, under Bayh-Dole, if the university and researcher choose not to patent, the government has the option of patenting. Bayh-Dole Act of 1980, Pub. L. No. 96-517, 94 Stat. 3015 (codified as amended at 35 U.S.C. §§ 200–11 (2006)). Whether a decision to release software under an open source license represents an unwarranted interference with the government's option remains an open question, at least in theory. In practice, however, we are unaware of any situation where a decision to release software under an open source license has interfered with an agency's desire to patent.

nonpatent-based modes of technology transfer.[201] Whether university patenting and licensing practices are currently different from practices in the late 1990s would be a valuable subject for further research. Even if filing practices have changed, however, the existing stock of software patents may represent a lucrative source of revenues when asserted against firms that have commercialized successfully. The extent to which such holdup strategies are successful may depend on how the *CSIRO* line of cases are resolved.

CONCLUSION

In this Article, we have undertaken the first systematic investigation of university software ownership and litigation. We found that software patents represent a significant, and growing, percentage of university patent holdings. Moreover, our qualitative results and research from a complementary econometric analysis, suggest that universities have followed a "one size fits all" approach: those with higher patent propensities in other fields also tend to patent more software. From a policy standpoint, this finding suggests a problem. Software—and particularly pure software—is likely to follow a different commercialization path than invention in other fields. Therefore, patenting and exclusive licensing of software may yield a higher proportion of situations where the exclusive licensee uses a patent to hold up an entity that has successfully commercialized without the need for an exclusive license. Moreover, even if the goal is not promoting commercialization *per se*, but promoting start-ups, exclusive patent licenses are not necessarily critical to that goal.

To be sure, university practices with respect to software are not uniform. For example, universities that have policies friendly to open source are less likely to patent pure software. More generally, a fair number of universities—including universities with large numbers of software patents—may now be moving away from patenting and exclusive licensing of software.[202] However, the lingering impact of a substantial university patent portfolio, some of which has been licensed exclusively, may continue to be felt in socially unproductive litigation.

201. *See supra* notes 185–89 and accompanying text.

202. To investigate this proposition further, we are currently doing work investigating more recent patent applications that have resulted in issued patents.

APPENDIX

Questions for Technology Transfer Officers

1) Is there an official university policy regarding ownership and dissemination of software and software-based inventions? If yes, what is it?

2) Are there more informal guidelines or norms that govern ownership and dissemination of software or software-based inventions? If yes, what are they?

3) How does the technology transfer office decide whether it should seek a patent on software or software-based inventions?

4) What is the university's policy, if any, towards open source software?

5) Are there any trends or changes in university policy towards ownership of software or software-based inventions that are worthy of note?

6) Have you had any conflicts with faculty members regarding how software or software-based inventions should be owned or disseminated? In cases of conflict, who makes the determination?

Questions for Academics/Graduate Students

1) Are you aware of an official university policy regarding ownership and dissemination of software and software-based inventions? If yes, what is it?

2) Are you aware of informal guidelines or norms that govern ownership and dissemination of software or software-based inventions? If yes, what are they?

3) Do you know how the technology transfer office decides whether it should seek a patent on software or software-based inventions?

4) Are you aware of any university policy towards open source software? If yes, what is it?

5) Are there any trends or changes in university policy towards ownership of software that are worthy of note?

6) Have you had any conflicts with the technology transfer office regarding how software or software-based inventions should be owned or disseminated? In cases of conflict, who makes the determination?

7) Do you produce or use open source software?

8) Do you generally disclose software or software-based inventions to the technology transfer office?

Table A1: Comparison of Bessen-Hunt Keyword-Based Classification of Software Patents with Our Classification

	Our Classification		
Bessen-Hunt Classification	Non SW	SW	Total
Non SW	2,325	202	2,527
	92.01	7.99	100
	91.32	51.01	85.89
SW	221	194	415
	53.25	46.75	100
	8.68	48.99	14.11
Total	2,546	396	2,942
	86.54	13.46	100
	100	100	100

Table A2: Comparison of Graham-Mowery IPC-Based Classification of Software Patents with Our Classification

	Our Classification		
Graham-Mowery IPC Classification	Non SW	SW	Total
Non SW	2,544	341	2,885
	88.18	11.82	100
	99.92	86.11	98.06
SW	2	55	57
	3.51	96.49	100
	0.08	13.89	1.94
Total	2,546	396	2,942
	86.54	13.46	100
	100	100	100

Table A3: Comparison of Graham-Mowery USPC-Based Classification of Software Patents with Our Classification

	Our Classification		
Graham-Mowery USPC Classification	**Non SW**	**SW**	**Total**
Non SW	2,541	328	2,869
	88.57	11.43	100
	99.80	82.83	97.52
SW	5	68	73
	6.85	93.15	100
	0.20	17.17	2.48
Total	2,546	396	2,942
	86.54	13.46	100
	100	100	100