3-2007

An Empirical Look at Software Patents

James Bessen
Boston University School of Law

Robert M. Hunt

An Empirical Look at Software Patents

JAMES BESSEN
Research on Innovation
Boston University School of Law
jbessen@researchoninnovation.org

ROBERT M. HUNT
Federal Reserve Bank of Philadelphia
Ten Independence Mall
Philadelphia, PA 19106
bob.hunt@phil.frb.org

Software patents have grown rapidly in number and now comprise 15% of all patents. They are acquired primarily by large manufacturing firms in industries known for strategic patenting; only 5% belong to software publishers. The very large increase in software patent propensity over time is not adequately explained by changes in R&D investments, employment of computer programmers, or productivity growth. The residual increase in software patent propensity is consistent with a sizeable increase in the cost effectiveness of software patents during the 1990s, perhaps arising from changes in the application of patent law to computer software.

1. Introduction

Over the past three decades, the federal courts, and to a lesser extent the US Patent and Trademark Office (USPTO), gradually shifted from a view hostile to patents on software-related inventions to a view that is now significantly more favorable. The traditional view in patent law is that abstract ideas are not in themselves patentable. Yet computer software can be quite abstract, sometimes being little more than an algorithmic

This paper presents some of the results from our 2004 working paper of the same name. Thanks to Peter Bessen of May8Software for providing a software agent to acquire our patent database and Annette Fratantaro for her work with the Compustat data set. Also thanks to John Allison, Tony Breitzman, Joseph Bush, and Rosemarie Ziedonis for sharing data with us. We thank the editor and referees, Iain Cockburn, Mary Daly, Dan Elfenbein, Terry Fisher, Bronwyn Hall, Joachim Henkel, Brian Kahin, David Mowery, Leonard Nakamura, Cecil Quillen, Eric von Hippel, Rosemarie Ziedonis, and numerous seminar participants for helpful comments. We alone are responsible for any remaining errors. The views expressed here are those of the authors and do not necessarily represent the views of the Federal Reserve Bank of Philadelphia or the Federal Reserve System.

representation of basic mathematical principles. Mindful of this, during the 1970s, the Patent Office and the courts applied very restrictive rules toward granting and upholding patents that included computer programs. Over time these restrictions were relaxed, gradually at first, after the Supreme Court decision in *Diamond v. Diehr* (450 US 175) in 1981, and then more rapidly after the decision in *In re Alappat* (33 F. 3d 1526) in 1994.

This paper explores the effect of this economic experiment on the patenting behavior of public firms. These changes may have been important to the software publishing industry, which experienced rapid growth at a time when software patent policy was still quite restrictive. But we also look at firms in other industries, because the legal changes have allowed these firms to acquire broad, relatively abstract patents by claiming software elements.

We begin by constructing our own definition of a software patent (there is no official definition) and assemble a comprehensive database of all such patents. In Section 1 we describe this process and the process of matching these patents to firm data in the Compustat database. In Section 2 we summarize the general characteristics of these data. We find that over 20000 software patents are now granted each year, comprising about 15% of all patents. Compared with other patents, software patents are more likely to be assigned to firms, especially larger US firms, than to individuals. They are also more likely to have US inventors. Surprisingly, most software patents are assigned to manufacturing firms and relatively few are assigned to firms in the software publishing industry (SIC 7372). Most software patents are acquired by firms in industries that are known to accumulate large patent portfolios and to pursue patents for strategic reasons (computers, electrical equipment, and instruments). These large inter-industry differences remain even after we control for R&D, software development effort, and other factors.

In Section 3 we perform regressions that explore the "propensity to patent" software inventions. This builds on the model of Hall and Ziedonis (2001) which, in turn, builds on the empirical literature of "patent production functions" (including Scherer, 1965; Bound et al., 1984; Pakes and Griliches, 1984; Griliches, Hall, and Hausman, 1986). We find a dramatic growth in software patent propensity even after controlling for R&D, employment of computer programmers, and other factors. This growth is quite similar to the remarkable growth in patent propensity that Hall and Ziedonis (2001) found in the semiconductor industry. The pattern of the residual increase is consistent with the explanation that changes in patent law made software patents significantly more cost effective, although there may be other explanations as well.

We also find that industries known for strategic patenting have much higher patent propensities. Section 4 concludes the paper.

To date, there have been relatively few empirical analyses of software patents. Allison and Lemley (2000) report on characteristics of software patents in a study of all patents. In addition, Allison and Tiller (2003) discuss Internet business method patents which are a subset of software patents. Graham and Mowery (2003) offer some preliminary results for the software publishing and computer systems industries. Graham and Mowery (2004) examine the prosecution of software patent applications.

2. Background and Data

2.1 Changing Legal Treatment and Strategic Patenting Industries

From the beginning, US patent law prohibited patents on abstract ideas, scientific discoveries, and natural laws. The Constitution calls for patents to promote the progress of the "useful arts," generally understood to mean industrial arts as distinct from the "liberal arts," which include science, mathematics, and the humanities. Courts have often argued that abstract ideas are not by themselves patentable, only their application to useful industrial processes. For example, in *O'Reilly v. Morse* (56 US 62, 1854), the Supreme Court upheld Morse's concrete claims that covered the invention of the telegraph, but rejected his abstract claim for "the use of the motive power of the electric or galvanic current . . . , however developed, for making or printing intelligible characters, letters, or signs, at any distances."

Several rationales have been advanced for restricting the patenting of abstract ideas. In *Morse*, the Supreme Court argued that such abstract claims would be so broad as to pose an obstacle to future inventors. In addition, this broad claim would allow Morse to take advantage of newly developed technologies without patenting them. Others have argued that standards of patentability including novelty and nonobviousness cannot be properly applied to abstract claims.[1]

In addition to the case law, two of the requirements of patentability in the patent statute reflect this concern about abstract ideas: Section 101 limits the subjects that can be patented and Section 112 requires that the invention be described in sufficient detail so that a person having ordinary skill in the art can make or use all of the embodiments of the claimed invention (Morse's patent failed to do this).

1. See, for example, Judge Archer's dissent in *In re Alappat* (33 F. 3d 1526, 1994).

As computer use grew, the Patent Office, the courts, and industry recognized that patents involving software might run afoul of these various strictures against patenting abstract ideas, given the potentially abstract nature of software. In 1965, President Johnson appointed a commission that ultimately recommended against patent protection for computer programs. Initial policies at the Patent Office and the Supreme Court decision in *Gottschalk v. Benson* (409 US 63, 1972) were seen as highly restrictive (Samuelson, 1990). But there was also significant disagreement between the various courts and the Patent Office, leading to uncertainty about exactly which computer-related inventions were patentable subject matter and how the other patentability requirements should be applied to these inventions.

These differences were resolved gradually. In 1981 (*Diamond v. Diehr*, 450 US 175), the Supreme Court clarified the applicability of patent protection for inventions involving software used in industrial processes.[2] Following this, the Patent Office liberalized its granting policy (Samuelson, 1990, pp. 1093–4). In *In re Alappat* (33 F. 3d 1526, 1994), the Federal Circuit held that subject matter limitations did not apply if the software produced "a useful, concrete, and tangible result," even if that result was just on a computer screen.

Several Federal Circuit decisions also largely eliminated the enablement requirement for software inventions (Burk, 2002; Burk and Lemley, 2002).[3] Reviewing case law, Burk and Lemley (2002, p. 1162) write: "For software patents, however, a series of recent Federal Circuit decisions has all but eliminated the enablement and best mode requirements. In recent years, the Federal Circuit has held that software patents need not disclose source or object code, flow charts, or detailed descriptions of the patented program. Rather, the court has found high-level functional description sufficient to satisfy both the enablement and best mode doctrines." The net effect of these various changes was that policy toward software patents became much less restrictive after 1981 and especially after the early 1990s.

Software patents were never prohibited in the US, as is sometimes claimed. In *Benson*, widely seen as the most restrictive decision, the Supreme Court stated: "It is said that the decision precludes a patent for any program servicing a computer. We do not so hold" (409 US 63, 71). In our analysis, we do find significant numbers of patents using software

2. The claimed invention used a computer program to govern an otherwise standard process for curing rubber.

3. See also Cohen and Lemley (2001) on the different treatment of software patents. In the words of an IBM patent attorney, "[the patent standard] currently being applied in the US invites the patenting of ideas that may have been visualized as desirable but have no foundation in terms of the research or development that may be required to enable their implementation" (Flynn, 2001).

were granted during the 1970s. Although the Presidential Commission and Congress debated whether this new technology should be covered by patents, in the courts and in the Patent Office, the central issue was the patenting of abstract ideas, not software technology *per se*. Subject matter requirements worked to limit the patenting of inventions using software, but this patenting was also restricted by enablement, novelty, and nonobviousness requirements, and changes in legal doctrine in all of these areas combined to dramatically ease the patenting of these inventions.

Nor were these changes in legal doctrine primarily concerned with "pure" software patents, that is, inventions that only claim elements involving software and general purpose digital computers. It is true that at certain times, the USPTO more readily granted patents that claimed industrial hardware combined with software. But the courts repeatedly held that patents could be obtained on software without such claims.[4] Conversely, the changes in legal doctrine had an important effect on the level of abstraction permitted in patents that claimed both software and industrial hardware or processes, including patents with "embedded" software. Patents involving software for controlling network routing, the functioning of microprocessors, printers and copiers, or analyzing seismic signals are all subject to varying degrees of abstraction and they all received varying treatment at the Patent Office and in the courts.

Indeed, some observers have assumed that software patenting was primarily a matter of concern to the prepackaged software industry. However, if the effect of these doctrinal changes was to allow more abstract claims in *industrial* patents, then they may also have had a particularly profound impact on manufacturing industries. Thus the economic experiment created by these changes in patent law concerns software patents both with and without industrial elements and it potentially concerns a wide range of industries. In our analysis below, we include a broad range of industries in addition to the software industry and we track the full range of patents that use software, not only those that run exclusively on general purpose computers.

Of course, these software-specific changes in patent law occurred against a backdrop of broader changes in patent law following the creation of a unified appeals court for patents suits in 1982 (see Hall and Ziedonis, 2001, for a nice summary). The court raised the evidentiary standards required to challenge patent validity and tended to broaden the interpretation of patent scope (Rai, 2003; Merges, 1997). The court

4. For example, *In re Freeman* (573 F. 2d 1273, 1978) applied a test where software more or less had to involve mathematical equations and "wholly pre-empt" the use of those equations for all possible applications in order to be unpatentable. In *In re Johnson* (589 F. 2d 1070, 1978), software that printed output was deemed to be patentable.

relaxed the standards for evaluating whether or not an invention is obvious to practitioners skilled in the art (Coolley, 1994; Dunner et al., 1995; Hunt, 1999; Lunney, 2001; Henry and Turner, 2005). The court was also more willing to grant preliminary injunctions to patentees (Cunningham, 1995; Lanjouw and Lerner, 2001) and to sustain large damage awards (Merges, 1997; Kortum and Lerner, 1999). The combined effect of these regulatory changes is that software patents appear to have gained greater appropriability and become less costly to obtain in absolute terms over time and also possibly relative to other patents.

The software industry was highly innovative and growing rapidly well before software patents became commonplace. Nominal investment in software grew 16% per annum during the 1980s (and 11% per annum during the 1990s, Grimm and Parker, 2000). This innovativeness is important for two reasons. First, in interpreting the results below, the growing use of software is an important factor. Second, given this history, it is not at all clear that patent protection was essential for innovation in this industry.

This is not unusual. When surveyed, American firms in a number of other innovative industries (including semiconductors and precision instruments) rate patents as a *relatively* less effective form of appropriability (Levin et al., 1987; Cohen, Nelson, and Walsh, 2000). Instead, they cite lead time advantages, learning curves, complementary sales and service, and secrecy as generally more important sources of appropriability.

Yet even in industries where patents are rated as ineffective, we sometimes observe that firms acquire large patent portfolios. We call these industries, including the computer, electrical equipment, and instruments industries, the "usual suspects." They are also found to account for a major share of the growth in patenting in recent years (Hall, 2003). Some researchers have suggested that firms in these industries may patent heavily in order to obtain strategic advantages, including advantages in negotiations, cross-licensing, blocking competitors, and preventing suits (Levin et al., 1987, fn 29, and Cohen, Nelson, and Walsh, 2000).

In principle, strategic patenting can arise whenever individual products involve many patentable inventions and the cost of obtaining patents is sufficiently low (see Bessen, 2003 and Hunt, 2006 for theoretical models). Firms may acquire large numbers of patents so that even if they have an unsuccessful product, they can hold up rivals, threatening litigation. Innovative firms may acquire "defensive" patent portfolios to make a credible counter-threat. The outcome may involve the cross-licensing of whole portfolios, where firms agree not to sue each other and those firms with weaker portfolios pay royalties (Grindley and Teece, 1997).

So, in what follows, it is natural for us to be alert to the possibility that software patents may also be acquired for strategic purposes and we will find distinctive behavior in the industries known for strategic patenting. An important implication of strategic patenting is that policy changes that "strengthen" patents (or make them cheaper to acquire) can lead to a kind of "Prisoner's Dilemma" game that actually *decreases* the private incentive to engage in R&D.

2.2 What Is a Software Patent?

How many software patents are being granted? Although the patent office maintains a system for classifying patents, this system does not distinguish whether the underlying technology is software or something else. Researchers must construct their own definitions.

As noted above, legal changes created a natural economic experiment that affected all patents that use software, not just "pure" software patents. Because we want to investigate the effects of this experiment, we need an operational definition of software patents that includes all these patents.

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.

2.2.1 Identifying Software Patents

How can we identify patents that fit this description? Griliches (1990) reviews the two main techniques that researchers have used to assign patents to an industry or technology field: (1) using the patent classification system developed by the patent office; and (2) reading and classifying individual patents. In this paper, we use a modification of the second technique.

We began by reading a random sample of patents, classifying them according to our definition of software, and identifying some common features of these patents. We used these to construct a search algorithm to identify patents that met our criteria. We used this algorithm to perform a keyword search of the US Patent Office database, which identified 130,650 software patents granted in the years 1976 to 1999.[5]

5. See the Appendix for a description of the search algorithm we use.

Next, we conducted several tests to validate the power and accuracy of our algorithm. These tests used three random samples of patents that were each read and classified as to whether they were software patents or not. The first sample consisted of 400 patents granted in the years 1996–98. The second sample, also from the 1990s, consists of 330 software patents identified by John Allison and used in Allison and Lemley (2000) and Allison and Tiller (2003).[6] The third sample consists of 100 patents from the 1980s that were identified as software patents by our algorithm.

First, we measured the power of our algorithm, that is, we measured the percentage of patents that our algorithm correctly identified as software patents in our two random samples (See Table AII in the Appendix).[7] Our algorithm found 78% of the software patents in our first random sample and 92% of the software patents in John Allison's sample.

Second, we measured the accuracy of our algorithm, that is, we measured the percentage of patents that were incorrectly identified as software patents by our algorithm. In our 1996–8 sample of patents our algorithm identified as software patents, 16% were not, based on our reading of the patents.[8] Among patents granted during the 1980s (our third sample) our algorithm identified as software patents, 9% were not, again based on our reading of the patents.

These low error rates may seem surprising, given that patent drafters sometimes obscure the use of software by making the language in the claims sound like special purpose industrial equipment. For example, patent 4931783 (from 1988) claims "a computer-controlled display system," for what is actually a computer program running on a standard computer.[9] But although the patent *claims* are obscure (the subject matter restrictions applied to the claims), the written description describes the best mode of the invention as a "computer program," so our algorithm successfully identifies this as a software patent. Indeed, patent drafters have several strong reasons to make clear the real nature of the invention in the written description: if they do not do so, they

6. Thanks to John Allison for sharing his data with us. The data used in Allison and Lemley (2000) are based on reading 1,000 randomly selected patents issued between mid 1996 and mid 1998. The data were augmented in Allison and Tiller (2003) by examining 2,800 patents issued between 1990 and 1999 identified via a keyword search (for the terms *Internet* or *world wide web*) restricted to patents included in classes 705, 707, or 709. Note that in the Allison and Tiller taxonomy, internet business method patents are a subset of software patents. Allison uses a similar definition of software patent, identified by whether the claims involve logic processing steps or not.

7. This is equivalent to calculating (1—the false negative rate).

8. The associated false positive rate in this sample was 2%—among the 346 patents we manually classified as *not* software, the algorithm classified 8 as software patents.

9. Thanks to Robert Barr for bringing this to our attention.

risk finding that their own product may not be covered. In addition, without such clear description, the patent may not be enabled or the best mode requirement may not be met.[10] Because our algorithm searches the written description for key terms, it is surprisingly accurate even for patents granted in the 1980s.

The validation tests suggest that our algorithm finds most of the software patents and it makes relatively few mistakes. It might be possible, however, that our algorithm makes more mistakes for certain groups of patents than others, so we compared the error rate across three pairs of subgroups: patents granted in the 1990s versus those granted in the 1980s; patents to US inventors versus those to foreign inventors; and patents in the "Computers and Communications" technology classes versus others (using the NBER class categories in Hall, Jaffe, and Trajtenberg, 2001a). Using *t*-tests, none of the differences in error rates were statistically significant at the 1% or 5% levels. Our algorithm had a somewhat higher error rate for patents outside the computers and communications technology classes (16% versus 7%), but this difference was only significant at the 10% level.

Although the algorithm does make errors, it performs reasonably well, and it seems unlikely that it introduces significant biases to our patent counts or regression coefficients. In the next section, we report on an additional robustness test. We compared our tabular and regression results with those obtained using an entirely different means of identifying software patents.

2.2.2 Using Patent Classes to Identify Software Inventions

Why did we use this rather laborious method rather than simply counting patents in certain patent classifications? First, in a longitudinal study patent classes are problematic because the classification system changes over time and the patent office continually re-classifies issued patents. Moreover, lawyers are known to draft patents so that they avoid falling into certain classes in order to influence the examiner's prior art search or some other aspects of the examination (Lerner, 2004, p. 19).

Second, economists have long recognized the poor correspondence between patent classes and economic concepts of industry or technology (see, for example, Schmookler, 1966; Scherer, 1982a; Soete, 1983; Scherer,

10. In order to be valid, a patent must be "enabled," that is, the written description must provide enough detail so that a person having ordinary skill in the art can practice the invention without undue experimentation or development. In addition, the written description must provide the "best mode" of practicing the invention known to the inventor at the time of filing.

1984; Griliches, 1990). Patent classes are not designed with social scientists in mind but are used primarily to aid prior art search.[11]

In the US classification system, there are no patent classes for software *per se*. Instead, software inventions are included in functional categories along with hardware inventions. For instance, one class includes "arrangements for producing a permanent visual representation of output data." This is a functional description that includes software programs, hardware computer displays, and even electric and mechanical signs that predate electronic computers.

Still, we did examine the efficacy of using the patent classification approach for identifying software inventions, as proposed in Graham and Mowery (2003). In that paper, the authors identified a number of subdivisions of the International Patent Classification system (IPC) where many patents assigned to large US software companies may be found.[12] We also ran our validation tests using the Graham and Mowery classification. The results, reported in the Appendix, show that our algorithm had substantially better power and accuracy. In addition, the IPC-based definition would exclude half of the patents obtained by the top 200 publicly traded software firms during the 1990s. In contrast, our definition accounts for about 4/5 of all patents obtained by the top 200 software firms in the 1990s.

Nevertheless, to check whether our main results were robust to our choice of algorithm, we ran most of our tabular analyses and regressions below using the Graham–Mowery classification of software patents. The results were broadly similar. For example, the distribution across industries was similar (see Appendix Table AIII). If anything, Graham and Mowery's software patents are more highly concentrated among computer and electronics firms (71% compared to 58%). We also ran our Poisson regressions from Table V using Graham–Mowery software patent counts as the dependent variable. Standard errors were somewhat higher, but the coefficients were not markedly different.

To summarize, our algorithm finds most of the software patents, it makes relatively few errors, and these errors do not seem to vary

11. Allison and Lemley (2000) and Allison and Tiller (2003) also reject the idea of using patent classifications to identify software patents. According to Griliches (1990), a patent class is "based primarily on technological and functional principles and is only rarely related to economists' notions of products or well-defined industries (which may be a mirage anyway). A subclass dealing with the dispensing of liquids contains both a patent for a water pistol and for a holy water dispenser. Another subclass relating to the dispensing of solids contains patents on both manure spreaders and toothpaste tubes" (Griliches, 1990, p. 1666).

12. The subdivisions include G06F 3/, 5/, 7/, 9/, 11/, 13/, and 15/; G06K 9/, and 15/; and H04L 9/.

systematically across subgroups.[13] Our main results hold even when we use a very different methodology for identifying software patents.

2.3 The Matched Sample

We also explore the characteristics of the firms that obtained software patents. To do this, we matched a large portion of both software patents and other patents to firms in the 1999 vintage of the Compustat database.

Our main population of interest consists of US-owned public firms that perform R&D. This group performs a large share of domestic R&D and it should provide a relatively stable group for comparison over time. But it does limit the relevance of our conclusions to this group, however, and so our analysis has little to say about start-up firms, individuals, universities, etc.[14]

We begin by matching our patents to firms (that is, the assignees) using the NBER Patent Citations Data File (Hall, Jaffe, and Trajtenberg, 2001a).[15] This data set matches patents to the 1989 vintage of firms contained in Compustat, so we do a variety of things to supplement those matches:

1. We added the largest 25 publicly traded software firms ranked by sales (only one of which is included in the NBER file).[16]
2. We merged the data set with a set of firm-patent matches provided to us by CHI Research.[17] Those data encompass most of the significant patenting firms (public or private) over the past 25 years.
3. Using data contained in Compustat, we identified 100 of the largest R&D performers in 1999 that were not already included in our data set. We matched these firms and their subsidiaries to their patents using a keyword search on the USPTO web site.

13. For examples of errors see Hahn and Wallsten (2003), who critiqued an earlier version of our working paper.

14. Descriptive statistics for the sample of firms (with and without patents) drawn from Compustat are reported in Columns 4 and 5 of Table AI.

15. To be precise, we match patent numbers in our data set with those found in the NBER data set. Where available, we use the firm CUSIP assigned by NBER to obtain financial data from Compustat.

16. As a result, we over-sample the software industry. According to BLS data, SIC 7372 makes up 0.2% of total private employment. SIC 7372 comprises 0.7% of employment among Compustat firms and 1.1% of employment among matched firms.

17. We again match by patent number and use the firm CUSIP assigned by CHI. Details on CHI's proprietary data is described at http://www.chiresearch.com/information/customdata/patdata.php3. We are grateful to Tony Breitzman for sharing the data with us.

The final file held 4,792 distinct subsidiaries and 2,043 parent firms from 1980 to 1999, an improvement of 1,230 subsidiaries and 305 firms over the NBER database for the same period.[18]

To test the coverage of this matched sample, we compared it to the target population in Compustat (that is, all US firms that publicly report financials with positive R&D). The matched sample performed 91% of the deflated R&D in the Compustat file over this period and accounted for 89% of the deflated sales by R&D-performing firms. Moreover, the coverage ratios are roughly constant over the entire sample period, varying only a few percentage points in each direction over 2 decades. Over this period, the matched sample also accounts for 68% of all successful US patent applications by domestic nongovernment organizations (mostly corporations) and 73% of software patents granted to these organizations. These coverage ratios were also quite stable over the 2 decades.[19]

However, only 37% of the R&D-performing firms contained in Compustat are matched to their patents in our data set and this coverage declined over the sample period as an increasing number of small firms have gone public since 1980. Thus this matched sample is broadly representative of the firms that perform most of the R&D and obtain the majority of patents, but it is not representative of entrants and very small firms.

2.4 Other Data

We used the NBER Patent Citations Data File (Hall, Jaffe, and Trajtenberg, 2001a) to obtain data on citations received and numbers of claims. To obtain data on employment of programmers and engineers, we used the Occupational Employment Survey conducted by the BLS. This source provides detailed occupational employment of three-digit SIC industries. Because not all industries were covered in all years of the survey during the early years, we linearly interpolated some employment shares.[20]

18. The original NBER sample accounted for 47% of the successful patent applications to US nongovernment organizations; our sample accounts for 68% of these patents.

19. The top ten assignees of software patents not included in the matched sample are, in rank order, Canon, NEC, Ricoh, Minolta, US Navy, Brother, Sharp, Samsung, Toyota, and the University of California.

20. Thanks to Joseph Bush of the BLS for providing the data. The employment categories we used for programmers were occupation codes 25102, systems analysts, and 25105, computer programmers; for engineers, we used 22100 (a group code). Because computer support occupations (25103 and 25104) were lumped in with the other codes during early years of the survey, we make a proportional adjustment in those years, reducing the employment counts of programmers by their relative share in the first year in which all categories were reported.

3. Summary Statistics

Table I reports the number of software patents and other patents granted per year and also the number of applications per year, conditional on the applications successfully resulting in a grant by the end of 1999. As can be seen, their numbers have grown dramatically in absolute terms and also relative to other patents. Today almost 15% of all patents granted are software patents.

Table I.
Number of Software Patents

	Patents Issued			Successful Patent Applications	
	Software Patents	Total Utility Patents	Software/ Total (%)	Software Patents	Total Utility Patents
1976	765	70,226	1.1	853	65,804
1977	884	65,269	1.4	1,094	65,978
1978	897	66,102	1.4	1,170	65,601
1979	795	48,854	1.6	1,439	65,726
1980	1,080	61,819	1.7	1,633	66,491
1981	1,275	65,771	1.9	1,821	63,910
1982	1,402	57,888	2.4	2,233	65,009
1983	1,443	56,860	2.5	2,297	61,563
1984	1,939	67,200	2.9	2,641	67,071
1985	2,453	71,661	3.4	2,924	71,442
1986	2,657	70,860	3.7	3,482	75,088
1987	3,530	82,952	4.3	4,055	81,458
1988	3,495	77,924	4.5	4,841	90,134
1989	4,974	95,537	5.2	5,755	96,077
1990	4,704	90,364	5.2	6,471	99,254
1991	5,347	96,513	5.5	7,091	100,016
1992	5,862	97,444	6.0	8,149	103,307
1993	6,756	98,342	6.9	9,459	106,848
1994	8,031	101,676	7.9	12,251	120,380
1995	9,000	101,419	8.9	16,617	137,661
1996	11,359	109,645	10.4	17,085	131,450
1997	12,262	111,983	10.9	13,087	114,881
1998	19,355	147,519	13.1		
1999	20,385	153,486	13.3		
2000	21,065	157,595	13.4		
2001	23,406	166,158	14.1		
2002	24,891	167,438	14.9		

Note: Utility patents excluding reissues. Successful patent applications are the number of patent applications that resulted in patent grants by the end of 1999.

Table II.
Characteristics of Software Patents (1990–95)

	Software Patents	Other Patents
Assignee type		
Nongov't. org. (firm)	88%	80%
Individual/unassigned	11%	18%
Government	2%	2%
US assignee (if assigned)	70%	51%
US inventor	69%	53%
Mean citations received	9.7	4.6
Number of claims	16.8	12.6
Percent of self-citations	12%	13%
Percent of patents owned by top 5% of assignees	63%	63%

Note: Total patents: 39,700 software, 546,058 other. Self-citations are the average of the upper and lower bounds (see Hall, Jaffe, and Trajtenberg, 2001a). Differences between the means in the first two columns are all significant at the 1% level.

3.1 Who Owns Software Patents?

Table II shows characteristics of software patents compared with other patents, using data from the NBER patent database. Software patents are more likely to be owned by firms than by individuals or government. They are also more likely to be owned by US assignees and to have US inventors.[21] After the US, the top countries ranked by inventors are Japan (18%), Germany (3%), Great Britain (2%), and Canada (2%).

Table III reports several measures of the size of the owners of patents in our matched sample by the type of patent (software or other).[22] Ranking patents by the size of their owners, the table shows that the median software patent has a larger owner than the median nonsoftware patent. This is also true for rankings by market value, sales, and R&D. Note that the unit of observation here is the patent, not the firm, so that firms with multiple patents are represented multiple times, and possibly, in both columns. Software patents are also slightly more likely to be obtained by newly public firms. Allison and Lemley (2000) also find that software patents are more likely to be obtained by larger entities as classified by the patent office.

21. Allison and Lemley (2000) find that their sample of software patents has about the average share of US inventors, although they use a somewhat different method to classify inventors' national origins.

22. A patent granted in a given year is associated with a (deflated) measure of size for the firm in that year.

Table III.

Firm Characteristics by Patent Type (1990–99)

	Software Patents	Nonsoftware Patents
Median firm market value (million $96)	24,485	11,554
Median firm sales (million $96)	13,382	8,940
Median firm R&D (million $96)	956	376
Newly public firm	5.8%	5.1%

Note: Table shows the characteristics of the owner of the median patent (patents ranked by owner size) for each type of patent during the years 1990–99 from the sample matched to Compustat. Firms that obtain both software and other patents are included in both columns of the table. Firms that don't obtain software patent appear in column 3 only. Covers 48,072 software patents and 274,529 nonsoftware patents. Newly public firms first appeared in the Compustat file within the last 5 years.

3.2 The Distribution of Software Patents Across Industries

Table IV shows the industries of the firms obtaining software patents in the sample matched to Compustat. Most of the software patents are obtained by manufacturing firms, especially in the electronics and machinery industries, which include computers. Software publishers (SIC 7372) acquire only 5% of the patents in this sample, and other software service firms, excluding IBM, account for 2%.[23] The "usual suspects" for strategic patenting—SIC 35, 36, 38 and IBM—account for 68% of software patents.

The distribution of software patents across industries appears to reflect something other than the *creation* of software. Columns 2 and 3 include two measures that reflect software creation: the share of programmers and systems analysts employed in the industry and the share of programmers, systems analysts, and engineers. These are the occupations most directly involved in the creation of software and so these shares should represent the relative software development effort, the first measure more narrowly than the second.

The manufacturing sector acquires 75% of software patents but employs only 11% of programmers and analysts (32% of software writers if engineers are included). Software publishing and services (SIC 737, including IBM) acquires only 13% of software patents but employs 33% of programmers and analysts (18% if engineers are included). There is little reason to expect software developers employed in software companies or finance or retailing to be far less productive than software developers

23. IBM accounts for 13% of the software patents in our sample and it is consistently the largest software patentee. We break it out separately, as it is not representative of the software services industry.

TABLE IV.
SUCCESSFUL SOFTWARE PATENT APPLICATIONS BY INDUSTRY (1994–97)

	Share of All					Software
	Software Patents (%)	Prog. (%)	Prog. + Engineers (%)	All Patents (%)	All Patents/ R&D	Patent Propensity (Table 5.2)
	(1)	(2)	(3)	(4)	(5)	(6)
Manufacturing	75	11	32	88	3.8	
Chemicals (SIC 28)	5	1	2	15	2.5	1.5
Machinery (SIC 35)	24	3	7	17	4.2	4.4
Electronics (SIC 36)	28	2	7	27	6.8	9.6
Instruments (SIC 38)	9	1	4	11	7.1	8.7
Other manu.	9	5	13	18	2.3	1.9
Nonmanufacturing	25	89	68	12	3.0	
Software publishers (SIC 7372)	5	} 33	18	1	1.0	1.0 (all SIC 73)
Other software (SIC 737 exc. 7372, IBM)	2			1	2.8	
Other nonmanufacturing	4	55	49	4	3.4	3.8
Addendum: IBM	6	–	–	2	5.0	

Note: covers 28,268 software patents and 148,552 total patents for firms in the sample of software patents matched by CUSIP to firms in Compustat. The numbers are for patent applications that resulted in an issued patent by 1999. Programmer employment is the share of total employment accounted for by computer programmers and systems analysts in 1997 from the Occupational Employment Survey of the BLS (which includes government employees). The third column adds employment of engineers. For SIC 737 the combined employment shares also include IBM. The fifth column shows patents granted per $10 million of R&D in 1996 dollars. The last column is the exponential of the coefficient for the industry given in Table V, Column 3, normalized so that the patent propensity of SIC 73 equals 1. The last row shows IBM's patents as a portion of all patents (not just those in our matched sample).

in manufacturing. These large differences suggest that industries differ dramatically in the degree to which they seek patent protection for their software.

This disparity also appears in the fifth column, which reports the differences in the simplest measure of patent propensity, the ratio of all patents to R&D. Software publishing firms get only a quarter of the number of patents per dollar of R&D that other firms obtain. This corresponds to the views expressed by software publishing executives that software patents are of little value to them (USPTO, 1994).[24]

The sixth column displays a measure of *software* patent propensity derived from the regression analysis described below. This is normalized so that the software patent propensity of SIC 73 has a value of one. Overall, software patents are more likely to be obtained by larger firms, established firms, US firms, and firms in manufacturing (and IBM); they are less likely to be obtained by individuals, small firms, foreign firms, and software publishers.

4. The Rising Propensity to Patent Software

From 1987 to 1996 the number of successful software patent applications (granted by 1999) increased 16.0% per annum. This growth was greater than any of a number of yardsticks one might measure it against: during roughly the same period, real industrial R&D grew 4.4% per annum, employment in computer programming related occupations grew at a 7.1% rate, and real business spending on own-account and outsourced programming grew 7.4% per year. This growth occurred against a general background of rising patent propensity—the ratio of all domestic patent applications to real R&D—grew about 2.1% per year.[25]

We can identify several possible factors that might contribute to increased software patenting: growth in R&D generally, growth in that portion of R&D that uses software, greater productivity in software development, and changes in the cost-effectiveness of software patents from regulatory or other sources. To help sort out the roles played by these different factors, we use a "patent production function" model of

24. In addition, BEA analysis of software investment (Grimm and Parker, 2000) implies that about 30% of software is produced as packaged software, the primary product of firms in SIC 7372. Yet this industry acquires only 5% of software patents.

25. Total industrial R&D increased from $121 billion in 1988 to $164 billion in 1998 in 1996 dollars (NSF, 2003). The BLS Occupational Employment Survey estimates 904,430 employees in "computer scientists and related occupations" in 1987–89 and 1,839,760 in 1998. The BEA estimates $38 billion in 1996 dollars for business spending on own-account and custom software in 1989 and $64 billion in 1998 (Grimm and Parker, 2000). The general increase in patent propensity has been explored by Kortum and Lerner (1999) and Hall and Ziedonis (2001).

Hall and Ziedonis (2001). This production function relates the number of successful patent applications made by a firm each year to its size, relative R&D spending and other characteristics, plus a time dummy, which serves to capture residual changes in patent propensity.

4.1 Specification

In the base specification of this model, the expected number of patents for firm i in year t, conditional on firm characteristics for that firm and year, is

$$E[n_{it}] = \exp\left(\alpha_t + \beta_1 \ln \text{Employees}_{it} + \beta_2 \ln \frac{\text{R\&D}_{it}}{\text{Employee}_{it}} + \beta_3 \ln \frac{\text{Capital}_{it}}{\text{Employee}_{it}} + \beta_4 \delta_i\right), \tag{1}$$

where δ is a dummy variable that equals one if the firm is a new entrant and zero otherwise. The right-hand side variables capture the effects of scale, R&D intensity, capital intensity, and new entrant status. The time dummy then captures changes in the propensity to patent. Differences in the time dummies between two different years correspond to log differences in the expected number of patents. This is our initial specification to which we add additional controls, including firm effects and a measure of software development intensity.

This equation can be interpreted as a "software patent production function," where the right-hand-side variables are input factors (scaled by employment). In this base specification, we use aggregate R&D, which includes R&D spent on software engineers, on hardware engineers, and on other specific R&D inputs. We do not observe these disaggregated factors, but we want to control for possible differences in their elasticities. We assume that firms within an industry face common prices for these inputs, firms are cost minimizers, and any firm heterogeneity in the productivity of software patent production is captured in the firm fixed effects. In the regressions, we control for variation in the use of these different factors by including the shares of software and other engineers in industry employment (to capture industry differences in the cost and use of these factors) and firm fixed effects to control for firm heterogeneity.

Because the left-hand-side variable in (1) is a count and many observations are zero, the equation can be estimated with a Poisson regression, as is frequently done in the literature. The Poisson regression assumes that the variance equals the expected value of the left-hand-side variable, but often Poisson specifications fail to meet this assumption and show "over-dispersion." Tests using a negative binomial specification reveal over-dispersion in our data too, and, following Hall and Ziedonis (p. 113), we take this as an indication to use heteroscedastic-consistent standard errors. Regressions using the negative binomial specification produced broadly similar estimates of the coefficients. However, as Hall and Ziedonis point out, negative binomial estimates are inconsistent if the true distribution is not actually negative binomial. The Poisson estimates, on the other hand, will still be consistent even with overdispersion, so we prefer to present Poisson estimates using heteroscedasticity-robust standard errors.

Our analysis differs from Hall and Ziedonis in two important ways, however. First, our dependent variable is not all of the patent applications of the firm, just software patent applications. In principle, this should cause no problem—one still expects size, R&D intensity, etc. to affect the level of software patents. But we may also want to control for the degree to which the firm directs resources to software development, which we do using employment shares. Second, Hall and Ziedonis study a narrowly defined industry whereas we study a broad range of industries. For this reason, we will want to control for industry or firm effects in some of our regressions.

In our base specification n_{it} is the number of software patent applications by firm i in year t that resulted in a patent granted by 1999. Because patent prosecution typically takes two years or so, we conduct our regressions through 1997.[26] The other variables are as follows: employees is the number of employees listed in Compustat in thousands, R&D is deflated by the GDP deflator, and capital is property, plant, and equipment deflated by the NIPA capital goods deflator. The "new firm" dummy is equal to 1 for the first five years a firm appears in Compustat.

4.2 Results

Column 1 of Table V shows the base regression. Column 2 adds seven industry dummies and variables to capture the relative use of programming personnel. The first of these variables is the ratio of "computer scientists and related occupations" to total industry employment in the BLS Occupational Employment Survey (OES) for the firm's SIC three-digit industry (or two-digit if Compustat assigns the firm only two digits). The second variable is the comparable share of engineers in total employment. The OES was conducted on a three-year cycle until 1995,

26. The mean lag from application to grant for software patents from 1980–95 was 2.3 years, suggesting at worst a modest truncation bias in 1997.

Table V.
The Propensity to Patent Software (Dependent Variable: Annual Number of Successful Software Patents Applications)

	1	2	3	4	5	Annual Contribution
Ln employment	0.88 (0.03)*	0.88 (0.03)*	0.88 (0.03)*	0.88 (0.03)*	0.62 (0.02)*	0.5%
Ln R&D/Emp.	1.01 (0.05)*	0.77 (0.06)*	0.77 (0.06)*	0.78 (0.05)*	0.22 (0.02)*	1.1%
No R&D dummy	−1.00 (0.17)*	−0.47 (0.18)*	−0.49 (0.18)*	−0.16 (0.19)	0.35 (0.12)*	−0.1%
Ln Capital/Emp.	−0.08 (0.04)	0.37 (0.05)*	0.37 (0.05)*	0.26 (0.05)*	0.27 (0.03)*	1.8%
Programmers/Emp.		12.44 (3.13)*	12.75 (3.09)*	7.21 (0.84)*	13.32 (0.75)*	1.2%
Engineers/Emp.		4.13 (0.55)*	4.05 (0.55)*	6.57 (0.44)*	6.06 (0.58)*	1.2%
New firm dummy	0.19 (0.13)	0.03 (0.13)	0.12 (0.15)	0.11 (0.13)		
New firm × SIC 73			−0.90 (0.33)*			
Other patents/Emp.				0.16 (0.02)*		
Industry dummies						
SIC 28		−5.58 (0.32)*	−5.52 (0.31)*			
SIC 35		−4.50 (0.25)*	−4.45 (0.25)*			
SIC 36		−3.73 (0.24)*	−3.68 (0.23)*			
SIC 38		−3.83 (0.25)*	−3.77 (0.25)*			
Other manu.		−5.33 (0.25)*	−5.26 (0.25)*			
SIC 73		−5.99 (0.63)*	−5.96 (0.62)*			
Other nonmanu.		−4.65 (0.28)*	−4.58 (0.28)*			
Firm fixed effects					Yes	
Period estimated	80–97	87–97	87–97	87–97	87–97	
Annual growth rate of year dummies	10.3%	9.4%	9.5%	7.2%	10.8%	
No. observations	23,108	13,136	13,136	13,136	6,587	
Log Likelihood	−68,548	−42,963	−42,870	−48,528	−9,659	

Note: Heteroscedastic-consistent standard errors in parentheses. All regressions include year dummies; Columns 1 and 4 include an intercept (not shown). Asterisk indicates significance at the 1% level. Regressions are Poisson regressions. Column 5 has firm fixed effects (Hausman, Hall, and Griliches, 1984). R&D is deflated by the GDP deflator, capital is property, plant, and equipment deflated by the NIPA capital goods deflator, and employment is in thousands. The "new firm" dummy is equal to 1 for the first five years a firm appears in Compustat. Other patents/employees is the ratio of patents to employees for other firms in the same two-digit SIC industry for the given year. Annual growth rate of year dummies is year dummy for 1996 minus the year dummy for the earliest year divided by the number of years elapsed (no dummy is estimated for 1997). Annual contribution is the estimated coefficient times the difference in sample means for the variable between 1996 and 1987 divided by 9 (the number of years).

so values for this variable in intervening years are linearly interpolated. For comparability, we used OES data only from 1987 onward, so the second column covers a shorter interval.

As discussed above, these employment share variables capture the disparate effects of different R&D inputs. The large coefficient on the programmer employment share indicates that a dollar of R&D at a firm with relatively more programmers will, all else equal, generate relatively more software patents. Engineers also increase the software patent productivity of R&D, but not nearly as much.

The elasticity of the scale variable, employment, is similar to estimates in previous studies. Hausman, Hall, and Griliches (1984) obtained an elasticity of 0.87 on R&D. Hall and Ziedonis obtained an elasticity of employment of 0.85 when they included some firm controls. The coefficient of the R&D intensity variable is, however, much higher than in Hall and Ziedonis, although when we include firm fixed effects below, this difference largely disappears, suggesting that it is picking up unmeasured firm/industry heterogeneity. Capital intensity (in Column 2) is also significant and similar in magnitude to one estimate by Hall and Ziedonis, but smaller than another. Hall and Ziedonis interpret this coefficient as evidence that capital-intensive firms may patent more because they are subject to holdup by rivals who patent strategically. That is, this is evidence of "defensive patenting." Our result suggests this hypothesis might apply more generally.

4.2.1 New versus Old Firms

We find that the "new firm" dummy variable is not significant. In contrast, Hall and Ziedonis find that their dummy variable is highly significant and has a relatively large coefficient.[27] They suggest that their result arises from new semiconductor design firms that need patents to secure financing (see also Hall, 2003). These smaller firms do not have complementary manufacturing facilities that may provide another means of appropriability.

Although our sample may not be representative of the entire population of new firms, this regression does include a reasonably large sample of new firms (1308 observations during the first five years of 314 firms). Hence our different result suggests that either software patents

27. There is also a difference between the definition of our new firm variable and theirs. Hall and Ziedonis include all firms that entered Compustat after 1982. We include firms that entered in 1982 or later, but our dummy variable equals 1 only for the first five years of entry. Their initial interest was to compare whether incumbent firms in particular benefited from the creation of the Court of Appeals for the Federal Circuit. Our interest is whether entrants appear to find additional benefits from software patents, something Hall and Ziedonis explored specifically for entrant design firms with an additional dummy variable.

are not comparably useful for obtaining financing in general or this "vertical disintegration" strategy is not broadly relevant outside the semiconductor industry, or both.

We explored this issue further by interacting the new firm dummy variable with industry dummy variables. Only one of these interaction terms was statistically significant, the one for SIC 73, business services, which in our sample largely consists of software services and prepackaged software firms. Column 3 shows a regression with just this interaction term added. The coefficient is negative and statistically significant, suggesting that new software firms obtain *fewer* software patents than established firms.[28] This suggests that new software firms may not obtain the same benefit from patents as do new semiconductor firms.

4.2.2 Cross-Industry Variation in Software Patent Propensity

Returning to Column 2, the coefficients on the industry dummies indicate large inter-industry differences in software patent propensity even after controlling for industry employment of programmers and engineers. Given the exponential specification, (1), differences between industry coefficients correspond to differences in the log of software patent propensity. Column 6 of Table IV shows corresponding differences in software patent propensity itself (that is, the exponential of the coefficients), normalized so that the software patent propensity of SIC 73 equals 1 (this includes IBM). As can be seen, the differences are quite large, with SIC 36 and SIC 38 obtaining an order of magnitude more software patents, all else equal, and machinery, SIC 35, not too far behind. In general, the industries that have a high propensity to patent software also have a high patent propensity in general (see Column 5 of Table IV), although the differences in software patent propensity are larger. These are the "usual suspect" industries.

The magnitude of these differences and the known importance of strategic patenting in electronics and computers suggest that these results may be the outcome of strategic patenting. To test this idea further, Column 4 drops the industry dummies but includes instead a measure of the degree to which other firms in the industry patent. This variable is the number of all patents obtained by other firms in the same two-digit industry as the observed firm, divided by the employment of those firms. The positive and significant coefficient suggests that firms

28. We also ran these regressions using the total number of patents, not just software patents, as the dependent variable and obtained similar results (not shown). Graham and Mowery (2003) report that the software patent propensities of established software publishers rose over the 1990s, but for entrant firms (those founded after 1984) there was no discernable trend.

obtain more software patents in industries that patent more overall, all else equal. This result is consistent with "defensive" patenting, although it could also arise from other industry differences, such as large differences in alternative means of appropriability.

4.2.3 Unobserved Heterogeneity

Our seven industry dummy variables do not capture the full extent of inter-industry heterogeneity or other firm heterogeneity. This was confirmed by tests using the random effects and fixed effects Poisson models (Hausman, Hall, and Griliches, 1984). We then face the choice of better controlling for this unobserved heterogeneity via a random effects or a fixed effects model. A Hausman test rejects the null hypothesis that the random effects estimates are consistent; that is, the firm effects appear to be correlated with the coefficients ($P = 0.000$), indicating that fixed effects are preferred.

Column 5 presents the fixed effects Poisson regression. This is a conditional maximum likelihood regression where the likelihood of each observation is conditioned on the likelihood of the observed sum of software patents for each firm over all years in the panel. This is only calculated for firms that have software patents in at least one year. Because many firms obtain no software patents, the sample size is reduced considerably for this regression. With fixed effects, the scale elasticity is somewhat smaller and is consistent with earlier research (Hausman, Hall, and Griliches, 1984) and the R&D intensity coefficient is now much more in line with the estimates in Hall and Ziedonis (2001). Note that the time trend in this regression is nearly the same as what we found in the regressions without fixed effects.

4.3 Time Trends

4.3.1 Role of Observable Factors

We use the fixed effects specification to interpret the relative influence of various right-hand-side variables on software patenting trends over time. Because (1) is exponential in form, changes in variables times their coefficients affect the log of software patents and can be interpreted as growth rates. To calculate annual growth rates, we evaluate the mean of each variable in 1996 and subtract the mean in 1987. We then multiply this difference by the estimated coefficient from Column 5 and divide by 9 (the number of years elapsed) to obtain an annual contribution of the variable to the growth of software patents for our sample. The resulting estimates are shown to the right of Column 5. A similar calculation for the year dummies is shown for all columns in a separate row.

The annual growth of software patenting among the firms in this sample (which excludes firms without any software patents over this entire period) is 16.4%, slightly larger than the aggregate 16.0% reported above. The majority of this increase is captured by the contribution of the year dummies (10.8%). The next largest contribution is from greater capital intensity (1.8%) followed by the growth in programmer employment (1.2%), engineer employment (1.2%) and R&D intensity (1.1%).

Thus the majority of the growth in software patenting could not be attributed to any of these explicit controls and can be attributed, instead, to rapidly rising patent propensity. Note that this increase in patent propensity is quite close to the result obtained by Hall and Ziedonis (2001) for all patents in just the semiconductor industry. A comparable calculation on their preferred specification also results in a 10.8% annual growth in patent propensity from 1987–95.[29] But such growth rates are far from typical for total patenting in most industries.

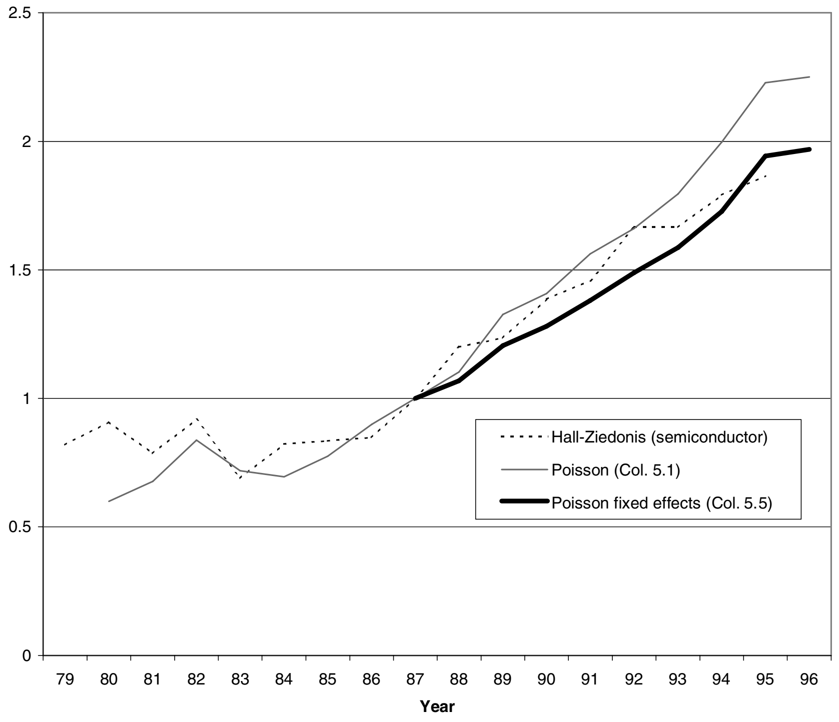
In Figure 1, we plot the year dummies from this regression (normalized to equal 1 in 1987), those from the Column 1 regression, which spans a longer time period, and the corresponding year dummies from Hall and Ziedonis' preferred specification. As can be seen, after the mid-1980s, all three series grow rapidly, persistently, and at about the same rate. Compared to the rate for 1987, and holding all other factors constant, firms were successfully applying for nearly 50% more software patents in 1991, and 164% more by 1996.

The close correspondence between the pattern of growth of software patent propensity and the growth of overall patent propensity in the semiconductor industry is striking. Only a minority of semiconductor industry patents were software patents (21% during 1994–97), so it appears that the patent propensity of nonsoftware patents in the semiconductor industry displayed essentially the same pattern of increase as software patents. This suggests that a common causal factor may have affected both software and nonsoftware patents in this industry, although the similarity could just be a coincidence.

4.3.2 Innovative Productivity or Legal Changes?

In principle, software patent propensity can be decomposed into two factors: an increase in the productivity of software developers, and change in the cost effectiveness of patenting.[30] The first factor means that software developers may produce more inventions using the same level of inputs. The second means that firms find it more attractive to patent a higher proportion of inventions, perhaps because the cost of

29. Thanks to Rosemarie Ziedonis for graciously sharing data.
30. See Hall and Ziedonis (2001) for a similar discussion.



Source: Hall and Ziedonis (2001) and Table V. Year dummies from Poisson regressions correspond to log relative patent propensity. Normalized to 1 in 1987.

FIGURE 1. YEAR DUMMIES, SOFTWARE PATENT PROPENSITY REGRESSIONS

doing so has fallen, or because such patents provide larger benefits than they did in years past, or both.

The most straightforward explanation consistent with our findings is that legal changes increased the cost effectiveness of software patents as described in Section 2. Eliminating the subject matter exclusion and reducing the nonobviousness and enablement requirements may have made software patents much easier (less costly) to obtain. Patents with more abstract claims may have had broader scope, increasing the appropriability each patent delivered. Both served to decrease the cost of appropriability.[31] The timing of these legal changes corresponds to

31. Another possibility is that alternatives to patents became less effective, increasing the relative cost effectiveness of software patents. But survey evidence suggests that trade secrecy did not decrease in importance and may have increased (Cohen, Nelson, and Walsh, 2000). We address changes in the efficacy of copyright protection below.

the observed increase in the software patent residual—beginning a few years after the creation of the CAFC and continuing to increase through the 1990s.

Moreover, strategic patenting may have amplified the effect of these changes. Reductions in the cost of appropriability may encourage firms to pursue more aggressive strategic behavior (Bessen, 2003; Hunt, 2006). This may, in turn, induce other firms to engage in defensive patenting, further increasing the patent propensity. Such behavior might explain why software patent propensity is highest among the "usual suspect" industries known for strategic patenting.[32] In addition, it would explain why the overall rise in software patent propensity closely tracks the rise in patent propensity among semiconductor firms. Finally, the importance of capital intensity is consistent with the strategic patenting reported in Hall and Ziedonis (2001), as described in Section 4.2.

On the other hand, an explanation based on increased programmer productivity faces several difficult obstacles. First, there is little in the IT literature or productivity literature that suggests any awareness of a large productivity increase. In order to explain the 10% per annum rise in patent propensity, the productivity increase must have been extremely large and one would expect that IT professionals and productivity economists would have been well aware of it. One candidate may be the wider use of software development tools. Although there was a strong market for software development tools, the actual contribution of these tools to productivity is widely debated and studies find that computer-aided software engineering tools often go unused (Kemerer, 1992). It is also true that software has been declining in price, but the BEA attributes most of the estimated price decline to growth in the market for software driven by rapidly falling prices for complementary computer hardware (Grimm and Parker, 2000, p. 6), *not* to greater programmer productivity.

Nevertheless, to test for the possibility that the trend in software patent propensity can be attributed to a persistent increase in productivity growth among programmers, we added to our fixed effects specification in Column 5 of Table V an interaction of the industry employment share of programmers with a dummy for years after 1991. The coefficient on this term is positive and statistically significant.[33] But the effect on the unexplained trend in software patent propensity is small; it falls to 9.5% a year from 10.8% reported in Table V.

We conclude that an account based on legal changes and strategic patenting provides a parsimonious explanation for the patterns

32. Again we are referring to firms in SICs 35, 36, and 38.

33. The coefficient is 3.7 (0.20). The coefficient on the employment share of programmers falls to 4.9 (0.89). We also included a similar interaction for the employment share of engineers, but it was not statistically significant.

observed in the data. Improvements in software developer productivity might also have played a role, but it explains relatively little of the rise in software patent propensity over time.

4.3.3 The Ebb and Flow of Copyright Protection

Another legal factor that might explain the large increase in software patent propensity is diminishing protection for computer programs provided by copyright (see Graham and Mowery, 2003). Lemley and O'Brien (1997) describe the rise and fall of the "nonliteral infringement" doctrine. Established in the 1986 decision in *Whelan Associates v. Jaslow Dental Laboratories*, it gave copyright holders some protection over the features of their software programs. But the doctrine was rejected in the 1992 decision in *Computer Associates International v. Altai*.[34]

It is possible that the changing extent of copyright protection might explain the rise in software patent propensity. We informally interviewed several practicing IP attorneys who suggested that although *Whelan* may have led more firms to push for expanded patent protection of software, they did not think that copyright concerns had much influence on patenting behavior because a firm could obtain both copyrights and patents on its software. Nevertheless, copyright may have had some effect at the *margin*, that is, for software inventions that were just marginally worth patenting.

If the alternative of copyright protection substantially affected firms' propensity to obtain patents, then one might expect a decrease in patent propensity after 1986, less growth between 1986 and 1992, and then an increased growth rate after 1992. No significant fluctuation is observed in Figure 1. To test this further, Table VI shows regressions using the fixed effects specification of Table V, Column 5. We have replaced the year dummies with a time trend and we interacted the time trend with a dummy variable that is one from 1986 through 1992. We also repeat this regression from 1984–1997 without the employment share data. We do find a statistically significant lower rate of patent propensity growth during the *Whelan* period in both specifications. However, the effect is not economically very significant, amounting to less than 5% of the trend rate of growth. This is consistent with the interpretation that copyright concerns may have exerted an influence on software patenting at the margin, but that this is not a major explanatory factor. Finally, note that the industries that patent most heavily also tend to use embedded software, so copyright protection was less relevant to them.

34. 797 F.2d 1222 (3rd Cir 1986) and 982 F.2d 693 (2d Cir 1992), respectively.

Table VI.
Test for a Break 1986–92 (Dependent Variable: Annual Number of Successful Software Patents Applications)

	1	2
Ln employment	0.57 (0.02)*	0.44 (0.01)*
Ln R&D/emp.	0.21 (0.02)*	0.23 (0.02)*
No R&D dummy	0.32 (0.11)*	0.04 (0.09)
Ln Capital/emp.	0.35 (0.03)*	0.24 (0.02)*
Programmers/emp.	11.75 (0.68)*	
Engineers/emp.	10.77 (0.51)*	
Year	0.047 (0.004)*	0.108 (0.002)*
Year $*$ ($86 \leq$ year ≤ 92)	−0.002 (0.000)*	−0.001(0.000)*
Period estimated	87–97	84–97
No. observations	6,587	8,692
Log likelihood	−10,094	−13,221

Note: Heteroscedastic-consistent standard errors in parentheses. Asterisk indicates significance at the 1% level. Regressions are Poisson regressions. All regressions are Poisson regressions with firm fixed effects (Hausman, Hall, and Griliches, 1984). R&D is deflated by the GDP deflator, capital is property, plant, and equipment deflated by the NIPA capital goods deflator, and employment is in thousands.

5. Conclusion

This paper documents a dramatic increase in software patenting, and software patent propensity, over time. Much of this growth cannot be explained by growth in aggregate investment in software, R&D, employment of computer programmers or engineers, or growth in the productivity of writing new programs. The timing of the surge in software patenting is consistent with many legal changes that have made these patents easier to obtain. In short, it appears that software patents have become relatively more cost effective over time.

The rapid growth in software patents and software patent propensity is not a phenomenon dominated by the software publishing industry. In fact, it occurs primarily among a group of industries, including computers, electronics, and instruments. Other researchers have established the role of strategic patenting in these industries and strategic patenting does provide a parsimonious explanation for many of our results. According to this theory, software patents are significant because they provide a cost effective way for firms to build strategic patent portfolios. If this interpretation is correct, the growth in software patents may not be associated with an improvement in the incentives to innovate, particularly in the "usual suspect" industries. But this remains a topic for future research.[35]

35. We provide some preliminary indications in our working paper of the same name (Bessen and Hunt, 2004) and we continue to develop those results.

Appendix: Search Algorithm

The search query used is:

((“software” in specification) OR (“computer” AND “program” in specification))

AND (utility patent excluding reissues)

ANDNOT (“chip” OR “semiconductor” OR “bus” OR “circuit” OR “circuitry” in title) ANDNOT (“antigen” OR “antigenic” OR “chromatography” in specification)

Table AI.
Summary Statistics

	Table V Sample		Total R&D Performing Sample	
	Mean	Median	Mean	Median
Annual successful patent applications	25.9	1		
Software patent applications	3.6	0		
Firm sales (mill. $96)	2,632.7	375.1	1,032.0	41.6
Firm R&D (mill. $96)	100.2	11.7	36.8	2.8
Percent not reporting R&D	27.9%		–	
Firm employees (1000s)	13.3	2.4		
Percent new firms	10.0%		37.2%	
No. of observations	13,136		28,581	
No. of firms	1,443		4,559	

Note: The total R&D sample corresponding to Table V consists of all Compustat observations for US firms with nonmissing R&D. Sales and R&D are deflated using the GDP deflator. New firms are firms during the first five years in which they appeared in Compustat.

Table AII.
Validation Tests of Software Patent Selection

	Software Patent Selection Method	
Test Sample	Bessen–Hunt (%)	Graham–Mowery (%)
Power*		
Bessen–Hunt 1996–98	78	26
Allison	92	46

Continued

Table AII.
CONTINUED

Test Sample	Software Patent Selection Method: Bessen–Hunt (%)	Software Patent Selection Method: Graham–Mowery (%)
Accuracy**		
Bessen–Hunt 1996–98	16	30
Bessen–Hunt 1980–89	9	

*Percent of patents identified as software patents via manual examination correctly identified as a software patent by the selection method.
**Percent of patents identified as software patents by selection method that are not software patents, based on manual examination of the patents.
The Bessen–Hunt 1996–98 test sample consists of 400 randomly selected patents from those years, of which 50 were identified as software patents, but 54 were true software patents. The Bessen–Hunt 1980–89 sample consists of 100 randomly selected patents that were selected as software patents by our algorithm. The Allison test sample consists of 330 software patents identified by John Allison (Allison and Lemley, 2000; Allison and Tiller, 2003). Allison read each patent, identifying these as patents claiming steps involving logic processing.

Table AIII.
Industry Distribution of Software Patents Using Graham–Mowery Definition

	Bessen–Hunt Software Patents (Table 4.1) (%)	Graham–Mowery Software Patents (%)
Manufacturing	75	73
Chemicals (SIC 28)	5	3
Machinery (SIC 35)	24	26
Electronics (SIC 36)	28	33
Instruments (SIC 38)	9	6
Other manu.	9	4
Nonmanufacturing	25	27
Software publishers (SIC 7372)	5	5
Other software (SIC 737 exc. 7372, IBM)	2	2
Other nonmanufacturing	4	2
Addendum: IBM	6	12

See notes accompanying Table IV.

References

Allison, J.R. and M.A. Lemley, 2000, "Who's Patenting What? An Empirical Exploration of Patent Prosecution," *Vanderbilt Law Review*, 58, 2099–2148.

——, ——, K.A. Moore, and R. Derek Trunkey, 2003, "Valuable Patents," George Mason Law and Economics Research Paper No. 03-31.

——, and E.H. Tiller, 2003, "Internet Business Method Patents," in W.M. Cohen and S.A. Merrill, eds., *Patents in the Knowledge-Based Economy*, National Research Council, Washington, DC: National Academies Press, 259–284.

Bessen, J., 2003, "Patent Thickets: Strategic Patenting of Complex Technologies," Research on Innovation Working Paper.

——, and R.M. Hunt, 2004, "An Empirical Look at Software Patents," Federal Reserve Bank of Philadelphia Working Paper No. 03-17/R.

Bound, J., C. Cummins, Z. Griliches, B.H. Hall, and A. Jaffe, 1984, "Who Does R&D and Who Patents?" in G. Zvi, ed., *R&D, Patents, and Productivity*, Chicago: University of Chicago Press.

Burk, D.L., 2002, "Patent Disclosure Doctrines: Enablement and Written Description," *Public Hearings on Competition and Intellectual Property Law and Policy in the Knowledge-Based Economy*, Washington: Federal Trade Commission.

——, and M.A. Lemley, 2002, "Is Patent Law Technology Specific?" *Berkeley Technology Law Journal*, 17, 1155–1206.

Cohen, J. and M.A. Lemley, 2001, "Patent Scope and Innovation in the Software Industry," *California Law Review*, 89, 1–57.

Cohen, W.M., R.R. Nelson, and J.P. Walsh, 2000, "Protecting Their Intellectual Assets: Appropriability Conditions and Why US Manufacturing Firms Patent (or Not)," NBER Working Paper No. 7552.

Coolley, R.B., 1994, "The Status of Obviousness and How to Assert It as a Defense," *Journal of the Patent and Trademarks Office Society*, 76, 625–644.

Cunningham, M.A., 1995, "Preliminary Injunctive Relief in Patent Litigation," *IDEA*, 35, 213–259.

Dam, K.W., 1995, "Some Economic Considerations in the Intellectual Property Protection of Software," *Journal of Legal Studies*, 24, 321–377.

Dunner, D.R., J.M. Jakes, and J.D. Karceski, 1995, "A Statistical Look at the Federal Circuit's Patent Decisions; 1982–1994," *Federal Circuit Bar Journal*, 5, 151–180.

Flynn, J.D., 2001, "Comments on the International Effort to Harmonize the Substantive Requirements of Patent Laws" IBM submission at http://www.uspto.gov/web/offices/dcom/olia/harmonization/TAB42.pdf

Graham, S.J.H. and D.C. Mowery, 2003, "Intellectual Property Protection in the U.S. Software Industry," in W.M. Cohen and S.A. Merrill, eds., *Patents in the Knowledge-Based Economy*, National Research Council, Washington, DC: National Academies Press, 219–258.

—— and ——, 2004, "Submarines in Software: Continuations in U.S. Software Patenting in the 1980s and 1990s," *Economics of Innovation and New Technology*, 13, 443–456.

Griliches, Z., 1990, "Patent Statistics as Economic Indicators: A Survey," *Journal of Economic Literature*, 28, 1661–1707.

——, B.H. Hall, and J.A. Hausman, 1986, "Patents and R&D: Is There a Lag?" *International Economic Review*, 27, 265–283.

Grimm, B. and R. Parker, 2000, "Recognition of Business and Government Expenditures for Software as Investment: Methodology and Quantitative Impacts, 1959–98," Bureau of Economic Analysis, mimeo.

Grindley, P.C. and D.J. Teece, 1997, "Managing Intellectual Capital: Licensing and Cross-Licensing in Semiconductors and Electronics," *California Management Review*, 39, 8–41.

Hahn, R.W. and S. Wallsten, 2003, "A Review of Bessen and Hunt's Analysis of Software Patents," AEI-Brookings Joint Center, mimeo.

Hall, B.H., 2003, "Exploring the Patent Explosion," invited lecture at the ZEW Workshop on Empirical Economics of Innovation and Patenting, Mannheim, Germany.

——, A.B. Jaffe, and M. Trajtenberg, 2001a, "The NBER Patent Citations Data File: Lessons, Insights and Methodological Tools," NBER Working Paper No. 8498.

——, ——, and ——, 2001b, "Market Value and Patent Citations: A First Look," Working Paper No. E01-304, University of California at Berkeley Economics Department.

——, and R.H. Ziedonis, 2001, "The Patent Paradox Revisited: An Empirical Study of Patenting in the U.S. Semiconductor Industry, 1979–1995," *RAND Journal of Economics*, 32, 101–128.

Hausman, J.A., B.H. Hall, and Z. Griliches, 1984, "Econometric Models for Count Data with an Application to the Patents-R&D Relationship," *Econometrica*, 52, 909–938.

Henry, M. and J.L. Turner, 2005, "The Court of Appeals for the Federal Circuit's Impact on Patent Litigation," mimeo, Department of Economics, University of Georgia.

Hunt, R.M., 1999, "Patent Reform: A Mixed Blessing for the U.S. Economy?" Federal Reserve Bank of Philadelphia *Business Review* (November/December), 15–29.

——, 2001, "You Can Patent That? Are Patents on Computer Programs and Business Methods Good for the New Economy?" Federal Reserve Bank of Philadelphia *Business Review*, 1st Quarter, 5–15.

——, 2006, "When Do Patents Reduce R&D," *American Economic Review*, 96, 87–91.

Kemerer, C.F., 1992, "How the Learning Curve Affects CASE Tool Adoption," *IEEE Software*, 9, 23–28.

Kortum, S. and J. Lerner, 1999, "What Is Behind the Recent Surge in Patenting?" *Research Policy*, 28, 1–22.

Lanjouw, J.O. and J. Lerner, 2001, "Tilting the Table? The Use of Preliminary Injunctions," *Journal of Law and Economics*, 44, 573–603.

Lerner, J., 2004, "The New New Financial Thing: The Sources of Innovation Before and After *State Street*," NBER Working Paper No. 10223.

Lemley, M. and D.W. O'Brien, 1997, "Encouraging Software Re-Use," *Stanford Law Review*, 49, 255–304.

Levin, R.C., A.K. Klevorick, R.R. Nelson, and S.G. Winter, 1987, "Appropriating the Returns from Industrial Research and Development," *Brookings Papers on Economic Activity*, 3, 783–820.

Lunney, G.S., Jr., 2001, "E-Obviousness," *Michigan Telecommunications and Technology Law Review*, 7, 363–422.

Merges, R.P., 1997, *Patent Law and Policy, 2nd Ed*, Charlottesville, VA: Michie Law Publishers.

——, 1999, "As Many as Six Impossible Patents Before Breakfast: Property Rights for Business Concepts and Patent System Reform," *Berkeley Technology Law Journal*, 14, 577–615.

National Science Foundation, 2003, *Research and Development in Industry: 2000*, Arlington, VA: National Science Foundation, Division of Science Resources Statistics.

Pakes, A. and Z. Griliches, 1984, "Patents and R&D at the Firm Level: A First Look," *Economic Letters*, 5, 377–381.

Rai, A.K., 2003, "Engaging Facts and Policy: A Multi-Institutional Approach to Patent System Reform," *Columbia Law Review*, 106, 1035–1135.

Samuelson, P., 1990, "*Benson* Revisited: The Case Against Patent Protection For Algorithms and Other Computer Program-Related Inventions," *Emory Law Review*, 39, 1025–1154.

Scherer, F.M., 1965, "Firm Size, Market Structure, Opportunity, and the Output of Patented Inventions," *American Economic Review*, 55, 1097–1125.

——, 1982a, "The Office of Technology Assessment and Forecast Industry Concordance as a Means of Identifying Industry Technology Origins," *World Patent Information*, 4(1), 12–17.

——, 1982b, "Demand-Pull and Technological Invention: Schmookler Revisited," *Journal of Industrial Economics*, 30, 226–237.

——, 1984, "Using Linked Patent and R&D Data to Measure Interindustry Technology Flows," in Griliches, ed., *R&D Patents and* Productivity, Chicago: University of Chicago Press, 417–461.

Schmookler, J., 1966, *Invention and Economic Growth*, Cambridge, MA: Harvard University Press.

Soete, L., 1983, "Comments on the OTAF Concordance between the U.S. SIC and the US Patent Classification," mimeo.

USPTO, 1994, *Hearings on Software Patent Protection*, Washington, DC: United States Patent and Trademark Office, January-February 1994.