

**ATTACHMENT A TO RAI SENATE TESTIMONY**

**University Software Ownership: Technology Transfer or Business as Usual?**

**Arti K. Rai, John Allison, Bhaven Sampat, and Colin Crossman**

**Working Paper**

**September 2007**

## 1. INTRODUCTION

Software patents and university-owned patents represent two of the more controversial intellectual property developments of the past 25 years. Various scholars have quarreled with the breadth of software patent claims (Burk and Lemley 2003; Rai 2003). Some have also suggested that, given the allegedly poor quality of prior art documentation and patent examiner training in the area of software, many issued software patents are likely to be invalid (Lunney 2001). More generally, there is significant debate over the extent to which software patents are likely to foster innovation (Bessen and Hunt 2006). Because software products are often “complex” and may infringe many patents, some producers of end-product software are also unhappy. Large incumbent software firms like IBM and Microsoft have favored recent legislative and judicial efforts to make patents easier to challenge and injunctive relief, particularly by non-practicing patentees, more difficult to secure. The controversy over software patents has also been fueled by the rise, and arguable success, of the “open source” movement in software. Open source software developers eschew patents. Although they do rely on intellectual property in the form of copyright, they do so for purposes of enunciating licenses under which source code is made freely available.

In contrast with software patents, university patents have a long history. However, the scale of university patenting has increased substantially over the past 25 years, since the 1980 passage of the Bayh-Dole Act. While the legal question was sometimes murky prior to 1980, Bayh-Dole made it unequivocally clear that universities can patent federally funded research.

Assertive university patenting has attracted a fair amount of attention in both the scholarly and popular literature (Washburn 2005; Mowery et al. 2004). Some have gone so far as to suggest that universities may be patent “trolls” (Lemley 2006). The vast majority of analyses have focused, however, on patenting within the life sciences. (Azoulay et al. (2005); Murray and Stern (2005); Rai and Eisenberg (2003). This focus is hardly surprising, as the majority of university-owned patents (and the majority of large revenue generators) emerge from the life sciences (Rai and Eisenberg 2003; Mowery et al. 2004). Moreover, 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 – applies most clearly to life science areas like drug development. There, the conventional wisdom is 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.

In the case of publicly funded software, by contrast, the need for a patent and exclusive license to promote further development is less immediately apparent.<sup>1</sup> Although development costs are not uniformly low, they are likely to be low relative to those in the biopharmaceutical industry. 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 (Bonaccorsi and Rossi 2003).

Some recent events have suggested, however, that universities (or at least their exclusive licensees)<sup>2</sup> might in fact be interested in strong assertions of proprietary rights over software. In 2004, a district court upheld a \$520.6 million jury verdict in a patent infringement suit against Microsoft brought by the

---

<sup>1</sup> 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 exclusive licensing. Mowery et al. (2004, 158) discusses the manner in which Columbia’s DNA co-transformation technology was developed commercially without the need for exclusive licensing.

<sup>2</sup> University technology transfer officials argue that exclusive licensees generally initiate the lawsuits and that universities are brought along reluctantly. E-mail Communication from Rick Brandon, University of Michigan Office of Technology Transfer, 11/21/2006

## Attachment A – Rai Senate Testimony, Hearing on “The Role of Federally Funded Research in the Patent System”

University of California and its exclusive licensee, Eolas Technologies.<sup>3</sup> The software patent in question essentially covers interactive web browsing. Eolas, a one-person startup run by one of the University of California professors listed as an inventor on the patent, filed suit on the patent on February 2, 1999, three months after it issued. And the UC/Eolas lawsuit is not unique. In a number of cases, universities and/or their exclusive licensees have sued firms (often, large incumbent firms) that commercialized the academic invention without the need for exclusivity. (Rai et al. 2007) An IBM vice-president was recently quoted as saying that “[u]niversities have made life increasingly difficult to do research because of all the contractual issues around intellectual property.” (Lohr 2006)

Universities have, on occasion, had disputes not only with large software firms but also with open source software developers. In 2001, a group of university bioinformaticians petitioned the National Institutes of Health (“NIH”), asking the agency to mandate that publicly funded bioinformatics software be distributed under an open source model. At about the same time, reports emerged of individual professors facing resistance to their attempts to operate their labs under an open source model. (Benner 2002)

This paper represents the first systematic study of which we are aware of university software ownership.<sup>4</sup> We rely in part on a unique, hand-curated database of university software patents. Our quantitative analysis focuses on patents primarily because there is no comprehensive data on the extent to which copyright is asserted by universities.<sup>5</sup> This quantitative analysis is supplemented by interviews conducted with technology transfer officers (particularly at universities that own large numbers of software patents) as well as academic scientists prominent in the open source movement. Through these interviews, we draw out the policies that major university software patentees have not only with respect to patents but also with respect to software ownership more generally.

The combination of our quantitative and qualitative inquiry yields a number of interesting results. First, software patents represent a significant percentage of university patent holdings. Second, the factor that has had the biggest effect on software patenting is not R&D generally, or even computer science R&D in particular, but the overall “patent propensity” of the university – that is, the tendency of the university to seek patents. These findings are reinforced by our qualitative results. Our interviews show that some universities view software as similar to other inventions that are more “physical” in the traditional sense. This attitude is problematic, as software is likely to follow a different commercialization path than these more physical inventions.

### **2. UNIVERSITY SOFTWARE PATENTING: HISTORY AND METHODS OF IDENTIFYING**

We began by using the USPTO’s *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 critical shifts in the legal regime surrounding software produced at universities.<sup>6</sup> Over these two decades, university patenting

---

<sup>3</sup> On appeal, the Federal Circuit ordered a new trial concerning the validity of the patent. *Eolas Technologies Incorporated v. Microsoft Corp.*, 399 F.3d 1325 (Fed. Cir. 2005). On August 24, 2007, immediately before the re-trial was set to begin, Microsoft and Eolas reached a confidential settlement.

<sup>4</sup> One study that touches on some related issues is Agrawal and Henderson (2002). Agrawal and Henderson examine the patenting practices of the MIT electrical engineering/computer science faculty. 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).

<sup>5</sup> Copyright attaches as soon as the software is created. Because there is no requirement to register copyrights, it is difficult to know the total volume of university software protected by copyright. Technically speaking, it is “all” protected by copyright, but one cannot estimate how much of it exists.

<sup>6</sup> A methodology that sampled on 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 our purposes. If we assume an average patent pendency of about 2 years (which was the average pendency through the 1980s and 1990s), we can get a sense of the impact of particular decisions. Sampling on year of issuance was also an issue for our regression analyses, which assess rates of patenting against (inter alia) R&D expenditures. We addressed

## Attachment A – Rai Senate Testimony, Hearing on “The Role of Federally Funded Research in the Patent System”

increased dramatically, from 385 issued patents in 1982 to 2946 in 2002. Patent jurisprudence in the area of software also evolved considerably during this period. Because this evolution is closely related to the manner in which we define the term “software patent,” we lay it out in some detail below.

### 2.1 History of Software Patenting

In the 1970s, the dominant intellectual property regime for software was copyright, not patent. A 1972 Supreme Court case, *Gottschalk v. Benson*, had 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.<sup>7</sup> Several years later, in the 1976 Copyright Act, Congress expressly endorsed copyright as an appropriate protection regime for software.

The IP terrain shifted in the 1980s. In the 1981 decision *Diamond v. Diehr*, 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. The Court allowed the patent even though software implementation represented the only novel feature of the invention.

Through the 1980s, the Court of Appeals for the Federal Circuit followed a test similar to that enunciated in *Diehr*. Under this test, if an invention’s claims involved nothing more than an algorithm or series of algorithms, then the invention could not be patented. 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. The overall message to patent attorneys was that software could be patented, but it had to be claimed as something else.

As the patent option was becoming more attractive, copyright was becoming much less so. In the early 1990s, a series of decisions from regional appellate courts<sup>8</sup> made it clear that copyright covered primarily the literal source code of the program. Courts saw broader coverage as running contrary to the principle that copyright is supposed to cover only expression, not ideas or utilitarian functions.

Greater changes lay in store. In the 1994 case of *In re Alappat*, the Federal Circuit effectively eliminated the physicality limitation by holding that the patentable subject matter requirement could be met by showing that the software created a new machine – a “special purpose” computer – when it was executed. Presumably all software would produce such a special purpose computer and hence be patentable. Four years later the Federal Circuit’s 1998 decision in *State Street v. Signature Financial Group* explicitly rejected any special subject matter test for software, arguing that software (like all invention) is patentable so long as it produced a “useful” result. After *State Street*, there was no need for even the fig leaf of a “physical” machine or process.

### 2.2 Identifying University Software Patents

Given this history, it is perhaps not surprising that identifying “software” patents is very difficult. To our knowledge, there have been only a few significant efforts to identify a large data set of software patents.<sup>9</sup> An initial paper by Graham and Mowery (2003), which does not attempt to define the term

---

this issue by experimenting with R&D “stocks” from a variety of different time periods prior to patent issuance. All time frames yielded qualitatively similar results.

<sup>7</sup> 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. In general, “pure” software patents of the type at issue in *Gottschalk* are likely to be broader than patents covering software embedded in a particular machine. Indeed, the advent of pure software patents is one reason for complaints of undue patent breadth in this arena.

<sup>8</sup> While all patent appeals go to the Federal Circuit, appeals from copyright cases go through the ordinary process of appeal to the regional courts of appeal.

## Attachment A – Rai Senate Testimony, Hearing on “The Role of Federally Funded Research in the Patent System”

“software patent,” relies upon certain International Patent Classifications (IPCs),<sup>10</sup> limited to patents in those classes owned by large software firms. A more recent paper by Graham and Mowery (2005) uses particular U.S. patent classes,<sup>11</sup> once again limited to patents in those classes owned by large software firms. 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 is underinclusive, because large numbers of software patents are owned by other kinds of firms and software patents are found in many other classifications.

Another significant effort to identify a large set of software patents, by Bessen and Hunt (2006), 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”). Rejecting the use of patent classifications,<sup>12</sup> 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.<sup>13</sup>

Although the Bessen and Hunt definition of software patent is reasonable, there are so many pitfalls associated with using automated keyword searches to identify a large set of patents, . 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. Moreover, software is a critical part of inventions in so many far-flung fields that reliance on particular search terms produces a data set characterized by substantial percentages of both Type I and Type II errors.<sup>14</sup>

---

<sup>9</sup> We exclude from our discussion a recent paper by Cockburn and MacGarvie (2006), which focuses on patents held by firms in 27 specific software markets.

<sup>10</sup> GM use IPC classes G06F (subclasses 3,4,7,9,11,12,13, and 15), G06K (subclasses 9 and 15) and H04L (subclass 9).

<sup>11</sup> The U.S. patent classes are 343, 358, 382, 704, 707, 709, 710, 711, 713, 714, 715, and 717.

<sup>12</sup> Bessen & Hunt (2006, 10-11). The Bessen & 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 involve only off-the-shelf software—that is, the software must be at least novel in the sense of needing to be custom-coded, if not actually meeting the patent office standard for novelty.

*Id.* at 8.

<sup>13</sup> The keyword search algorithm initially identifies a set of patents that use the words “software,” “computer,” or “program” in the claims or specification. Patents within the set that contain the words “semiconductor,” “chip,” “circuit,” “circuitry,” or “bus” are then excluded as are patents that contain the words “antigen,” “antigenic” or “chromatography.”

<sup>14</sup> Bessen and Hunt (BH) also identify substantial degrees of over- and underinclusiveness in the data set generated by their keyword search. *Id.* at 9. Table 1 uses university patents issued in 2002 to compare the BH approach with our own approach. The two approaches yield an approximately similar number of total patents (396 patents using our approach vs. 415 using the BH approach). However, 51% of the patents our approach identifies as software are not

## Attachment A – Rai Senate Testimony, Hearing on “The Role of Federally Funded Research in the Patent System”

Our definition of a software patent is a *patent in which at least one claim element consists of data processing, regardless of whether the code carrying out that data processing is on a magnetic storage medium or embedded in a chip.*<sup>15</sup> Not only is it possible to apply the definition consistently, but it also captures the realities of claim drafting. It is common for all or most of the elements in a patent claim to cover the prior art, with only one or perhaps two elements covering the purportedly novel and nonobvious advance. One finds large numbers of patents owned by computer hardware makers, for example, the claims of which initially read as though they cover something like 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 an algorithm. Some of this may be a consequence of the fact that, prior to *Alappat*, software had to be claimed not as a new algorithm *per se* but instead as part of a piece of hardware that allegedly did something different because of the new algorithm.

In a large set of patents it is impossible as a practical matter to target perfectly those patents in which only the software element (as contrasted with other elements of the claimed invention) is novel and non-obvious. But the fact that, under our definition, data processing must be identified in a claim element does suggest that software was seen by the claim drafter as important for novelty and non-obviousness. Otherwise the drafter would not have narrowed the claim by including the data processing limitation.

For this reason, our data set is quite likely to include those patents – and only those patents – in which data processing is central.

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, and it is not limited to any particular physical embodiment. (These sorts of patents theoretically could have issued with any frequency only after the Federal Circuit’s 1994 decision in *In re Alappat*.)<sup>16</sup>

### **3 UNIVERSITY SOFTWARE PATENTING: QUANTITATIVE TRENDS AND DETERMINANTS**

#### **3.1 Descriptive Statistics**

---

identified as such by the BH algorithm. Moreover, we classify as non-software 53% of the patents that BH classify as software. Similarly, one recent study that used software experts to read a sample of the BH patents asserts that more than 50% represented Type II errors. Layne-Farrar (2005).

A recent working paper by Bronywn Hall and Megan MacGarvie (HM) (2006) identifies patents that fall within the PTO subclasses to which patents owned by fifteen large software firms are assigned. HM then take the union of their approach and the first Graham/Mowery approach and intersect it with the Bessen and Hunt approach.

<sup>15</sup> Allison also uses this definition in a collaboration with Ronald Mann and Abe Dunn (2007) that has involved reading every patent issued from Jan. 1, 1998 to Dec. 31, 2002 to the almost 1,000 firms that appeared in Software Magazine’s annual “Software 500” list at least once during that five-year period. 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.

<sup>16</sup> To get a sense of possible Type I and Type errors, we assessed how the two Graham and Mowery approaches classified the 2,942 university patents issued in 2002. As shown in Tables 2 and 3, very few of the patents classified by us as non-software 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 86% of the patents we classified as software. Similarly, the GM-PTO approach did not classify as software 82% of the patents we classified as software. The Bessen-Hunt approach yields approximately the same number of total patents as does ours (396 patents using our approach vs. 415 using the BH approach). However, 51% of the patents our approach identifies as software are not identified as such by the BH algorithm. Moreover, we classify as non-software 53% of the patents that BH classify as software. Similarly, one recent study that used software experts to read a sample of the BH patents asserts that more than 50% represented Type II errors. Layne-Farrar (2005).

Figure 1 shows that university software patenting increased more than ten-fold over the 1982-2002 period, from 37 patents in 1982 to 396 patents in 2002. Over this period, the “pure software” proportion of university software patents also increased dramatically, from 13 percent to 32 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. 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 9 percent in 1982 to 13 percent in 2002, as seen in Figure 2.

Table 1 lists the 15 universities receiving the most software patents in 2002.<sup>17</sup> Together, these 15 institutions accounted for 60 percent of all university software patents issued in that year. The top five institutions alone (MIT, the University of California, Stanford, Caltech, and the University of Texas) account for over a third (34.2 percent) of all university software patents. The top five software patentees also represent the top five university patentees overall in 2002. However, moving farther 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 (6<sup>th</sup> in software patenting/15<sup>th</sup> in overall patenting), Georgia Tech (8<sup>th</sup>/20<sup>th</sup>), Carnegie Mellon (9<sup>th</sup>/51<sup>st</sup>), the University of Rochester (12<sup>th</sup>/50<sup>th</sup>) and the University of Illinois (14<sup>th</sup>/28<sup>th</sup>) particularly stand out as institutions substantially more prominent in software patenting than overall patenting.

With respect to the patenting of “pure” software, Table 1 shows that the top three patenting institutions overall also rank among the top recipients of pure software patents (1st, 2nd, and 4th). In contrast, although Caltech and the University of Texas rank high in overall software patenting (4<sup>th</sup> and 5<sup>th</sup> respectively), they rank relatively low in the patenting of pure software (23<sup>rd</sup> and 41<sup>st</sup> 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 (6th, 3rd, 13<sup>th</sup>, 14<sup>th</sup>, and 11th respectively).

As these examples suggest, several factors may affect university software patenting. First, the amount of software-related research and development, and thus the output of software, may matter. Second, the size of the overall research enterprise may matter, since software can be developed in many parts of the university. Third, an individual university’s overall propensity to patent, i.e., the share of 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), may also affect software patenting.<sup>18</sup> Although there are undoubtedly other factors that affect software patenting – for example, attentiveness to the specific question of how software patenting is viewed in the technology licensing office or among faculty<sup>19</sup> – these are difficult to measure. Despite the difficulty, we examine some of these factors in our qualitative analyses.

### 3.2 Patent Production Functions

To examine the relative importance of 1) software-related R&D, 2) overall R&D, and 3) university propensity to patent, we estimated simple “patent production functions” relating software patent counts in 2002, 1997, 1992, 1987, and 1982 to characteristics of each of the 202 Carnegie universities in our

---

<sup>17</sup> 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. Additionally, our impressions from anecdotal evidence lead us to doubt that the ‘dot-com’ bubble led universities to acquire very many of the patents on software-implemented business models that issued in rapidly increasing numbers in the late 1990’s. As noted further below, we are currently investigating more systematically the question of whether university software patent filing patterns changed after the burst of the dot-com bubble.

<sup>18</sup> 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. (2005) find that in a study of 3884 life science researchers, the overall “patent stock” of the university where the researcher is employed has an effect on number of patents held by the life sciences researcher.

<sup>19</sup> In the life sciences context, for example, one study that looked at patenting activity for 3884 researchers found that having co-authors who patent has a positive effect on patenting behavior. Azoulay et al. (2005).

## Attachment A – Rai Senate Testimony, Hearing on “The Role of Federally Funded Research in the Patent System”

sample. Specifically, we collected data on total research expenditures and computer science research expenditures from the NSF’s *Survey of R&D Expenditures at Universities and Colleges*<sup>20</sup>, and aggregated these data to create R&D stocks for the 5 year period prior to patent issue.<sup>21</sup> We also collected data on the number of non-software patents each university was issued in each issue year.<sup>22</sup> To facilitate interpretation, we took natural logarithms of the independent variables.<sup>23</sup> Since the dependent variables are integer valued, we estimated negative binomial regressions relating software patents and pure software patents to research expenditures.<sup>24</sup>

Tables 2 through 6 show the main results from these simple cross-sectional regressions, for 2002, 1997, 1992, 1987, and 1982 respectively. In negative binomial models, coefficients on log-transformed variables can be interpreted as elasticities. Model 1 of Table 2 shows that, in 2002, both computer science R&D and overall R&D have a positive and statistically significant impact on software patenting, but that the elasticity of software patenting with respect to computer science R&D is much smaller. In particular, a 1 percent increase in computer science R&D implies a .11 percent increase in software patenting, but a 1 percent increase in non-computer science R&D implies a .84 percent increase in software patenting.

One explanation for this is that a substantial number of software patents result from non-computer science R&D, consistent with the argument, made by Graham and Mowery (2003), that software is likely to be produced across the university. Similarly, outside the university context, Bessen and Hunt (2006) argue that many software patents are held by manufacturing firms.

However, it is also likely that universities with more research simply have faculty who are more aware of patenting opportunities. Alternatively, they may have larger or more sophisticated technology transfer efforts. In either case, the net result could be higher propensities to patent for a given amount of software-related R&D. To assess this, Model 2 controls for the total number of non-software patents issued to the university in 2002. After controlling for total non-software patenting, the impact of the amount of non-computer science R&D on patenting drops dramatically, and becomes statistically insignificant. However, computer science R&D remains positive and statistically significant. Perhaps most notably, overall patenting has a large and statistically significant

---

<sup>20</sup> This survey is conducted annually by the NSF Division of Science Resources Statistics, and includes R&D expenditures from all sources of funding. Also helpful for our purposes is the fact that this survey breaks down funding by science and engineering field and by source of funding (federal and non-federal).

<sup>21</sup> Similar results obtain if we aggregate computer science and engineering R&D. Because we sampled on issue years, and there was an approximately 2 year lag between patent application and issue (during the 1980s and 1990s), as well as a lag of uncertain duration between research and patent application, choosing a precise lag period was difficult. Accordingly, we experimented with various windows. All specifications (available upon request) yielded qualitatively similar results.

<sup>22</sup> Using non-software patents as a measure of university patent propensity could skew results to the extent there were universities for which the majority of patents were software patents. In those cases, patent propensity might appear artificially low. However, one cannot use the same variable on both sides of the regression. Fortunately, software patents did not represent a majority of patenting activity for any Carnegie university. We did attempt to find data on an additional proxy for patent propensity: the size of the technology transfer office. However, we were able to find data on size for fewer than half of the universities in our sample.

<sup>23</sup> In some cases, one or more of the right-hand side variables was zero, and the natural log of zero is undefined. Accordingly, we took natural logs of \$1 plus R&D.

<sup>24</sup> 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.

## Attachment A – Rai Senate Testimony, Hearing on “The Role of Federally Funded Research in the Patent System”

impact on software patenting, even after controlling for overall and computer science R&D. While a 1% increase in computer science R&D yields a .097% increase in software patents, a 1% increase in non-software patenting yields a .91% increase in software patenting.

Models 3 and 4 in Table 2 show similar regressions with the number of “pure” software patents as the dependent variable. Perhaps not surprisingly, the amount of computer science R&D has a much larger impact on “pure” software patenting than on software patenting overall. In the first specification (Model 3), a 1 percent increase in computer science R&D implies a .59 percent increase in pure software patents, more than five times larger than the effect on all software patents. Controlling for the amount of non-software patenting (in Model 4) reduces this effect somewhat, to .52%, but it is still much larger than the corresponding estimate for all software patents. On the other hand, Model 4 also shows that overall non-software patenting (after controlling for non-software R&D) still has a greater effect on “pure” software patenting. A 1% increase in number of non-software patents yields a .78% increase in pure software patents.

Table 3 shows qualitatively similar results for the 1997 cohort. Perhaps not surprisingly (given the uncertain status of software patents), results on the importance of computer science R&D for pure software patents in the 1982-1992 cohorts (shown in Tables 4 through 6) are more mixed. However, with the exception of the 1982 cohort, overall patent propensity remains a statistically and qualitatively significant determination of patenting for both software and pure software.

We also 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. Table 7 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 non-software patenting, with a 1 percent increase in non-software patenting implying a .46 percent increase in software patenting. The corresponding [something is missing here—should it be “finding” or another word?] for pure software patenting is neither qualitatively nor statistically significant. Moreover, none of the R&D measures is 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.

### 4 UNIVERSITY OWNERSHIP POLICIES

To complement the quantitative analyses, we also interviewed technology transfer managers responsible for software at thirteen of the fifteen universities that received the most software patents in 2002. The interviews sought to identify their opinions on determinants of university software patenting.<sup>25</sup> We also asked them questions about non-patent mechanisms for dissemination of software inventions, including “open source”-based dissemination. Further, we conducted interviews with university professors and graduate students prominent in software development and in the open source software movement.

According to Gerald Barnett, formerly at University of Washington (now at UC Santa Cruz) university technology licensing offices (TTOs) that have a long history of patenting tend to see software, including pure software, through the lens they use for other inventions, particularly in the life sciences.<sup>26</sup> In the life sciences, established technology transfer offices have long generated revenue not only through exclusive licensing (the model contemplated in the legislative history of Bayh-Dole) but also through nonexclusive licensing (a model not necessarily contemplated by Bayh-Dole but one that can generate substantial revenue).

Barnett’s perspective is in accord with results from our interviews with officials at some major technology transfer offices. Lita Nelsen, director of MIT’s technology transfer office notes (speaking of pure software),

---

<sup>25</sup> We were able to obtain interview with officials at all technology transfer offices except the California Institute of Technology and Columbia University.

<sup>26</sup> Interview with Gerald Barnett, October 2, 2003.

## Attachment A – Rai Senate Testimony, Hearing on “The Role of Federally Funded Research in the Patent System”

[I]f 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*: is it worth investing time and, when appropriate, patent money to try to license the software out (either as simple end use license or to a distributor or startup company to improve and distribute it.)<sup>27</sup>

Barnett’s view is also in accord with our regression results, which indicate that overall university patenting propensity strongly influences both overall and pure software patenting. The regression results underscore what can be seen through a casual review of the descriptive statistics in Table 1. As noted earlier, the top 5 software patentees (MIT, UC, Stanford, Caltech, and University of Texas) are also the top 5 overall patentees. Notably, none of these five was in the top five in computer science R&D spending for the years 1996-1998.

However, of these five universities, two – Caltech and the University of Texas – do *not* have significant numbers of pure software patents. (Caltech and the University of Texas ranked 23<sup>rd</sup> and 41<sup>st</sup> respectively.) At the University of Texas, the reason for this dearth of pure software patents may be attributable to the view, explicitly adopted by UT technology transfer officials in response to complaints from computer science faculty, that faculty should be allowed freedom to share their work as they wish. Such freedom includes express permission to use the GNU General Public License (GPL), a “viral” open source license under which source code is open to all but those who redistribute the software must also make any modifications they have made to the source code available to others. This policy was adopted in the 1990s and would therefore have affected the number of pure software patents that issued in 2002.<sup>28</sup> 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,<sup>29</sup> 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,<sup>30</sup> most software is for internal, in-house use and is probably not even reported to the technology transfer office. For software that is disclosed to the university, the faculty researcher decides how the software should be exploited. The technology transfer office is not only familiar with open source but is apparently eager to use the viral GPL: this license allows for the future possibility of forking, with one fork continuing to be available without charge via the GPL and the other fork converted into a non-viral license for which corporate clients might be willing to pay.

In contrast, the positions of MIT and the University of California are less explicitly favorable to open source.<sup>31</sup> 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 official policy.<sup>32</sup> The University of California system has a highly complex set of positions on open source licensing. This complexity emerges in part from the quasi-federated structure of the UC system, within which policies towards copyright licensing (and hence open source licensing) are determined by the individual campus. While

---

<sup>27</sup> E-mail communication from Lita Nelsen, July 27, 2005 (emphasis added). According to Nelsen, software patents tend to be worth patenting when the primary value is in the algorithm.

<sup>28</sup> Interview with Georgia Harper, Office of General Counsel, University of Texas, February 26, 2004. 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 determines that such distribution is in the best interests of the University and the public. <http://www.utsystem.edu/ogc/intellectualproperty/swadmpol.htm>; <http://www.otc.utexas.edu/Industry/Software/process.jsp>

<sup>29</sup> The Caltech technology transfer staff refused requests for an interview.

<sup>30</sup> Interview with C. Titus Brown, August 17, 2005

<sup>31</sup> 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. Interview with Kathy Ku. However, Stanford has long had a policy allowing professors to put their inventions into the public domain if they so desire.

<sup>32</sup> Interview with Lita Nelsen, August 8, 2005.

## Attachment A – Rai Senate Testimony, Hearing on “The Role of Federally Funded Research in the Patent System”

some campuses, such as Berkeley and San Diego, are familiar with the open source model, other campuses are much less familiar with it.<sup>33</sup> Various important campuses, including UC Berkeley, appear to be in the midst of fine-tuning their policies. 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. (Benner 2002) At about the same time, UC Berkeley computational biologist Steven Brenner encountered difficulties in releasing his lab’s software under an open source license.<sup>34</sup> By 2004, however, UC Berkeley had announced a policy that appeared to give a much stronger 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.<sup>35</sup> Although it is not clear whether researchers have the final say in cases of conflict, the new policy clearly appears to be 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 and non-profit institutions, and made available for a fee for commercial use. In general, the new Berkeley policy appears to encourage the possibility of releasing software (and/or underlying source code) under different types of licenses for different purposes.

As noted earlier, five universities – University of Washington, Georgia Institute of Technology, Carnegie Mellon, the University of Rochester, and the University of Illinois – rank substantially higher in software/pure software patenting than in overall patenting. 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 (and digital products more generally). The University of Washington appears to be the only university in the U.S. with a technology licensing office specifically devoted to software. This office now has 7 full-time professionals as well as 2-4 part-time students. As for Georgia Tech, its level of computer science R&D funding relative to other R&D funding is quite high: it ranks 5<sup>th</sup> in computer science funding and only 32<sup>nd</sup> in other funding. Finally, Carnegie Mellon ranks 2<sup>nd</sup> in computer science funding and 87<sup>th</sup> in other funding. Carnegie Mellon is also home to the Software Engineering Institute (SEI), a research consortium founded by software firms. Patents from SEI are assigned to Carnegie Mellon, and the software firms in the consortium receive an automatic, royalty-free license.<sup>36</sup>

Notably, although the University of Washington, Georgia Tech, and Carnegie Mellon have a significant 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. At the University of Washington, 90% of the revenue brought in by its Digital Ventures office in FY 2005 came from copyright licensing. The Digital Ventures web site features a large variety of unpatented software that can be secured for commercial use through an on-line license with a standard rate.<sup>37</sup> Like UC-Berkeley, the University of Washington also encourages “forking” in its licenses, that is, licenses in which software (object code and/or source code) is made available free of charge for certain uses and available for a fee for other uses. Officials at Georgia Tech’s technology licensing office note 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 (that is, pure software patents) are often not very

---

<sup>33</sup> Interview with William Decker, UCSD (noting that Berkeley and San Diego often use open source licenses but that other campuses are not). Cf. Interview with Joel Kirschbaum, UCSF (noting that although his office does not use open source, they often make executable code available to academics free of charge)

<sup>34</sup> Interview with Steven Brenner, March 2004.

<sup>35</sup> Office of Technology Licensing, UC Berkeley, Frequently Asked Questions, <http://otl.berkeley.edu/about/faqs.php> (discussing open source options available on Berkeley’s software disclosure form).

<sup>36</sup> Interview with Carl Mahler, September 2, 2005

<sup>37</sup> [http://depts.washington.edu/ventures/UW\\_Technology/Express\\_Licenses/](http://depts.washington.edu/ventures/UW_Technology/Express_Licenses/)

## Attachment A – Rai Senate Testimony, Hearing on “The Role of Federally Funded Research in the Patent System”

valuable.<sup>38</sup> Finally, at Carnegie Mellon, the current technology transfer team states that it almost never patents pure software unless another entity (in their case, often SEI) is willing to pay for such patenting.<sup>39</sup>

Another interesting development that emerged from our interviews is a potential shift from patent to copyright protection of software developed at universities. While revenues can be generated from patented software inventions by means of non-exclusive licensing, universities may be reluctant to license non-exclusively because they must incur the immediate cost of obtaining a patent. In contrast, exclusive licensees typically pay patenting costs. Notably, however, software is unlike virtually every other university invention because patents do not have to serve as the foundation of a non-exclusive licensing scheme. Copyright protection, which attaches without cost upon creation of the software, will do the job. If software is to be commercialized, the copyright on it should be registered for practical reasons, but the cost of such registration is miniscule compared to the cost of obtaining a patent.<sup>40</sup> For this reason, at least some technology transfer offices say that they are beginning to shy away from seeking patents for the purpose of non-exclusive licensing. At the University of Washington, for example, the Digital Ventures office says that it takes a “hard look” at patenting and will use it only if there is a real need to provide monetary incentives for further improvement and commercialization of the technology (presumably by means of an exclusive license). The University of Washington’s Digital Ventures has also convinced startups to “go without a patent.”<sup>41</sup> Non-exclusive copyright licensing can be quite lucrative for universities. MINOS, a software package for solving large-scale optimization problems (linear and non linear programs) that is available through a non-exclusive copyright license, is one of Stanford’s largest money generators.<sup>42</sup> Moreover, as noted earlier, in FY 2005 the UW Digital Ventures office made 90% of its revenue from copyright licensing.

However, even nonexclusive copyright licensing with relatively low fees can be problematic for non-profit researchers. Thus, universities that want to balance the goal of academic access with that of revenue generation, such as the University of Washington, are assessing which licenses and royalty structures are appropriate for which situations. The technology transfer offices at the University of Washington, the University of California-Berkeley, and Stanford have all embraced the idea of dual licenses that give relatively inexpensive access to the non-profit sector but allow for revenue generation from the commercial sector. Indeed, the highly lucrative Stanford MINOS program is available via a dual copyright license: the commercial use license costs more than ten times as much as the academic use license.

Information technology can substantially facilitate nonexclusive licensing with low transaction costs. Universities such as Stanford already make software like MINOS available through simple Web-based interfaces.<sup>43</sup> Seven universities are currently participating in a pilot test of the Kauffman Foundation’s Web-based I-Bridge project, which aims to reduce the transaction costs associated with licensing. A significant percentage of the 739 projects available on the I-Bridge site appear to represent software – 56 included the term “software” in their title or brief description.<sup>44</sup> Many of these software projects are not patented and are available either free or by means of a dual license. Additionally, some large technology firms have decided to sponsor software projects at universities

---

<sup>38</sup> Interview with Kevin Wozniak.

<sup>39</sup> Interview with Carl Mahler.

<sup>40</sup> Copyright registration is not only required before an infringement action is filed, but timely registration also makes the recovery of substantial damages for infringement a far easier task.

<sup>41</sup> E-mail communication from Chuck Williams, September 2, 2005

<sup>42</sup> Interview with Kathy Ku

<sup>43</sup> [http://www.sbsi-sol-optimize.com/asp/sol\\_product\\_minos.htm](http://www.sbsi-sol-optimize.com/asp/sol_product_minos.htm)

<sup>44</sup>

<http://www.ibridgenetwork.org/searchresults.asp?sID=A7F8685984874659980E9D27E56A9D05&Page=1&RecsPerPage=10&Keyword=software&ignore=2&OrderBy=Rank&fbPR=&fbCOM=&Language=&ShowC=on&fbCHA=> (keyword search for “software”)

## Attachment A – Rai Senate Testimony, Hearing on “The Role of Federally Funded Research in the Patent System”

under an explicit “open collaboration” model. In 2005, four technology firms (IBM, Hewlett-Packard, Intel and Cisco) announced a set of “Open Collaborative Research” principles which under which certain types of sponsored research in software would be made freely available. IBM is now embarking on specific software projects at seven universities under the rubric of these principles (Lohr 2006).

The related question of using copyright to promote “open source” within the university is an interesting one. As we have noted, some universities have embraced open source, and these universities tend to have much smaller numbers of pure software patents. But even among technology transfer officials sympathetic to the goals of open source, a number of them mention difficulties that open source may create in the university setting. 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, don’t have to be shared with the university.<sup>45</sup> 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.<sup>46</sup> Additionally, software is often developed by groups, and TTOs 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).<sup>47</sup> TTOs are also wary that particular types of open source licenses will conflict with obligations to sponsors, including the federal government under Bayh-Dole.<sup>48</sup> 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.

Because our quantitative data conclude with patents originally filed in the late 1990s, it may be that universities have become less aggressive in patenting software in more recent years. 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 quite receptive to non-patent based modes of technology transfer. Whether university patenting and licensing practices are currently different from practices in the late 1990s would be a valuable subject for further research.<sup>49</sup>

### 5 Conclusions

Our results indicate that universities have become more active patentees of software, including pure software. They have clearly availed themselves of the opportunities afforded by Bayh-Dole and Federal Circuit decisions in the 1990s. Moreover, at least for our sample (which terminates with patents filed through the late 1990s), behavior with respect to total software patents and pure software patents is strongly affected by overall patent propensity. From a private point of view, this correlation may make sense, especially if the reason for the effect is (as Mowery and Sampat (2001) suggest) the existence of economies of scale at the technology transfer office. To the extent such scale economies exist, they are likely to lower the private marginal cost of patent acquisition. From a social point of view, however, patenting based on scale economies is problematic. Ideally, we would want decisions about whether to patent publicly-funded

---

<sup>45</sup> Memorandum from Pat Jones

<sup>46</sup> 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, which was made available to paying customers, would not be viral in character. The software producer MySQL has adopted this strategy. See <http://www.mysql.com/company/legal/licensing/commercial-license.html>

<sup>47</sup> Interviews with Chuck Williams; Dana Bostrom; Lita Nelsen

<sup>48</sup> In theory, under Bayh-Dole, if the university and researcher choose not to patent, the government has the option of patenting. 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.

<sup>49</sup> The authors of this article are currently undertaking this research.

## Attachment A – Rai Senate Testimony, Hearing on “The Role of Federally Funded Research in the Patent System”

academic research to be based not only on the private marginal costs of patent acquisition but on whether a patent is needed to facilitate commercialization in a specific case, which is likely to vary across inventions and fields. To put the point another way, a lack of differentiation between software and other research could be problematic. As case studies undertaken by Colyvas et al. (2001) point out, the optimal mode of university-industry technology transfer is likely to vary by industry and invention.<sup>50</sup> 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—does not apply neatly in software, where development costs often are relatively low.<sup>51</sup>

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 necessary for commercialization.<sup>52</sup> 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. 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, such as the Eolas case noted earlier, 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. In the Eolas case, it does not appear that Microsoft’s commercialization was aided by the activities of UC/Eolas. Rather, Microsoft and other firms began to use the browser technology at issue in the case well before the patent issued.<sup>53</sup> In this case, and others involving litigation over university software patents (Rai et al. 2007), commercialization by firms other than the university licensee was going forward, and patent rights/exclusive licenses were not necessary to facilitate “technology transfer.” Rather, contrary to the spirit of Bayh-Dole, software patents in these cases primarily allow universities to extract rents, and perhaps even to “hold up” development efforts. More generally, when software firms are arguing that development opportunities and university-industry collaborations are likely to be spurred through fewer, not more, university assertions of patent rights (Johnson 2007; Thursby & Thursby 2006), policies that treat patents as the default option for software are problematic.

While university software patenting grew dramatically over the period we studied, 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 number of software patents – may now be moving away from patenting and exclusive licensing of software.<sup>54</sup>

---

<sup>50</sup> 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 research, and consulting). Cohen, Nelson, and Walsh (2000). Even within the pharmaceutical industry, patents and licenses were less important than research publications and conferences.

<sup>51</sup> Of course, such costs may be somewhat higher in situations where the software in question is not “pure software.” But even in those cases, to the extent that the novel or non-obvious element is software, development costs are probably still low relative to the biopharmaceutical industry.

<sup>52</sup> The evidence generally cited for this argument is survey data presented in Jensen and Thursby (2001). The Jensen and Thursby survey of 62 technology transfer offices found that TTO managers thought that inventor involvement was often important in the commercialization of inventions. Presumably patents and licensing royalties represent the most efficient contractual mechanism for inducing transfer of tacit knowledge.

<sup>53</sup> According to the UC Berkeley web site on the Eolas case, Microsoft and other firms were selling the technology by the time the patent issued.

<sup>54</sup> To investigate this proposition further, we are investigating more recent patent applications that have resulted in issued patents.

Attachment A – Rai Senate Testimony, Hearing on “The Role of Federally Funded Research in the Patent System”

## REFERENCES

- Agrawal, Ajay, and Henderson, Rebecca, 2002. Putting Patents in Context: Exploring Knowledge Transfer from MIT. *Management Science*. 48:44-60.
- Allison, John R. & Lemley, Mark A., 2000. Who’s Patenting What? An Empirical Exploration of Patent Prosecution. *Vanderbilt Law Review*. 53: 2000-2174.
- Allison, John R. & Lemley, Mark A., 2002. The Growing Complexity of the Patent System. *Boston University Law Review*. 82: 77-144.
- Allison, John R. & Tiller, Emerson H., 2003. The Business Method Patent Myth. *Berkeley Technology Law Journal*. 18: 987-1084.
- Allison, John R., Lemley, Mark A., Moore, Kimberly A. & Trunkey, R. Derek, 2004. Valuable Patents. *Georgetown Law Journal*. 92: 435-476.
- Allison, John R., Dunn, Abe, and Mann, Ronald, 2007. Software Patents, Incumbents, and Entry. *Texas Law Review*. 85: 1579-1625.
- Azoulay, Pierre, Ding, Waverly and Stuart, Toby E., 2005. The Determinants of Faculty Patenting Behavior: Demographics or Opportunities? NBER Working Paper Series, Vol. w11348. Available at SSRN: <http://ssrn.com/abstract=727128>
- Benner, Jeffrey, 2002. Public Money, Private Code. *Salon*
- Bessen, James and Hunt, Robert (2006). An Empirical Look at Software Patents, *Journal of Economics and Management Strategy* (forthcoming)
- Bonaccorsi, Andrea and Rossi, Christina, 2003. Why Open Source Software Can Succeed. *Research Policy*. 32: 1243-1256.
- Borland, John, 2002. Once-Hot Net Business Crippled By Feuds, CNET News.com, August 2.
- Burk, Dan & Lemley, Mark, 2003. Policy Levers in Patent Law. *Virginia Law Review*. 89: 1575-1696.

Attachment A – Rai Senate Testimony, Hearing on “The Role of Federally Funded Research in the Patent System”

- Cockburn, Iain and MacGarvie, Meghan, 2006. Entry, Exit, and Patenting in the Software Industry, NBER Working Paper W12563.
- Cohen, Wesley, Nelson, Richard, and Walsh, John, 2000. Protecting Their Intellectual Assets: Appropriability Conditions and Why U.S. Manufacturing Firms Patent (or Not), NBER Working Paper No. 7752.
- Graham, Stuart J.H. & Mowery, David C., 2003. Intellectual Property Protection in the U.S. Software Industry, pp. \_\_\_\_, in *Patents in the Knowledge-Based Economy*, edited by Wesley M. Cohen & Stephen A Merrill. Washington D.C.: The National Academies Press.
- Graham, Stuart J.H. & Mowery, David C., 2005. Software Patents: Good News or Bad News, pp. \_\_\_\_ in *Intellectual Property in Frontier Industries: Software and Biotechnology*, edited by Robert Hahn. Washington D.C.: AEI-Brookings Press.
- Hall, Bronwyn and MacGarvie, Meghan, 2006. The Private Value of Software Patents, NBER Working Paper W12195.
- Johnson, Wayne, Congressional Testimony Before House Committee on Science and Technology, July 17, 2007
- Ku, Katherine, 2002. Software Licensing in the University Environment. *Computing Research News*.
- Lemley, Mark A., 2006. Are Universities Patent Trolls? (working paper).
- Lemley, Mark A. and Shapiro, Carl, 2006. Patent Holdup and Royalty Stacking (working paper).
- Lohr, Steve, 2006, December 14. IBM and Universities Plan Collaboration, *New York Times*. C11
- Mann, Ronald, 2005. Do Patents Facilitate Financing in the Software Industry? *Texas Law Review* 83:961-1030.
- Mann, Ronald and Sager, Thomas, 2006. Patents, Venture Capital, and Software Startups (working paper).
- Merges, Robert, 2006. Patents, Entry and Growth in the Software Industry (working paper).

Attachment A – Rai Senate Testimony, Hearing on “The Role of Federally Funded Research in the Patent System”

Mowery, David, C., Nelson, Richard R., Sampat, Bhaven N., and Ziedonis, Arvids A., 2004. *Ivory Tower and Industrial Innovation: University-Industry Technology Before and After the Bayh-Dole Act in the United States*. Palo Alto: Stanford University Press.

Mowery, David C. and Sampat, Bhaven N. 2001. Patenting and Licensing University Inventions: Lessons from the History of Research Corporation. *Industrial and Corporate Change*.

Murray, Fiona and Stern, Scott, 2005. Do Formal Intellectual Property Rights Hinder the Free Flow of Scientific Knowledge? An Empirical Test of the Anti-Commons Hypothesis. NBER Working Paper No. W11465. Available at SSRN: <http://ssrn.com/abstract=755701>

Owen-Smith, Jason and Powell, Walter W. *To Patent or Not: Faculty Decisions and Institutional Success at Technology Transfer* (working paper).

Rai, Arti K., 2003. Engaging Facts and Policy: A Multi-Institutional Approach to Patent System Reform. *Columbia Law Review*. 103: 1035-1135.

Rai, Arti K. & Eisenberg, Rebecca, 2003. Bayh-Dole Reform and the Progress of Medicine. *Law and Contemporary Problems*.

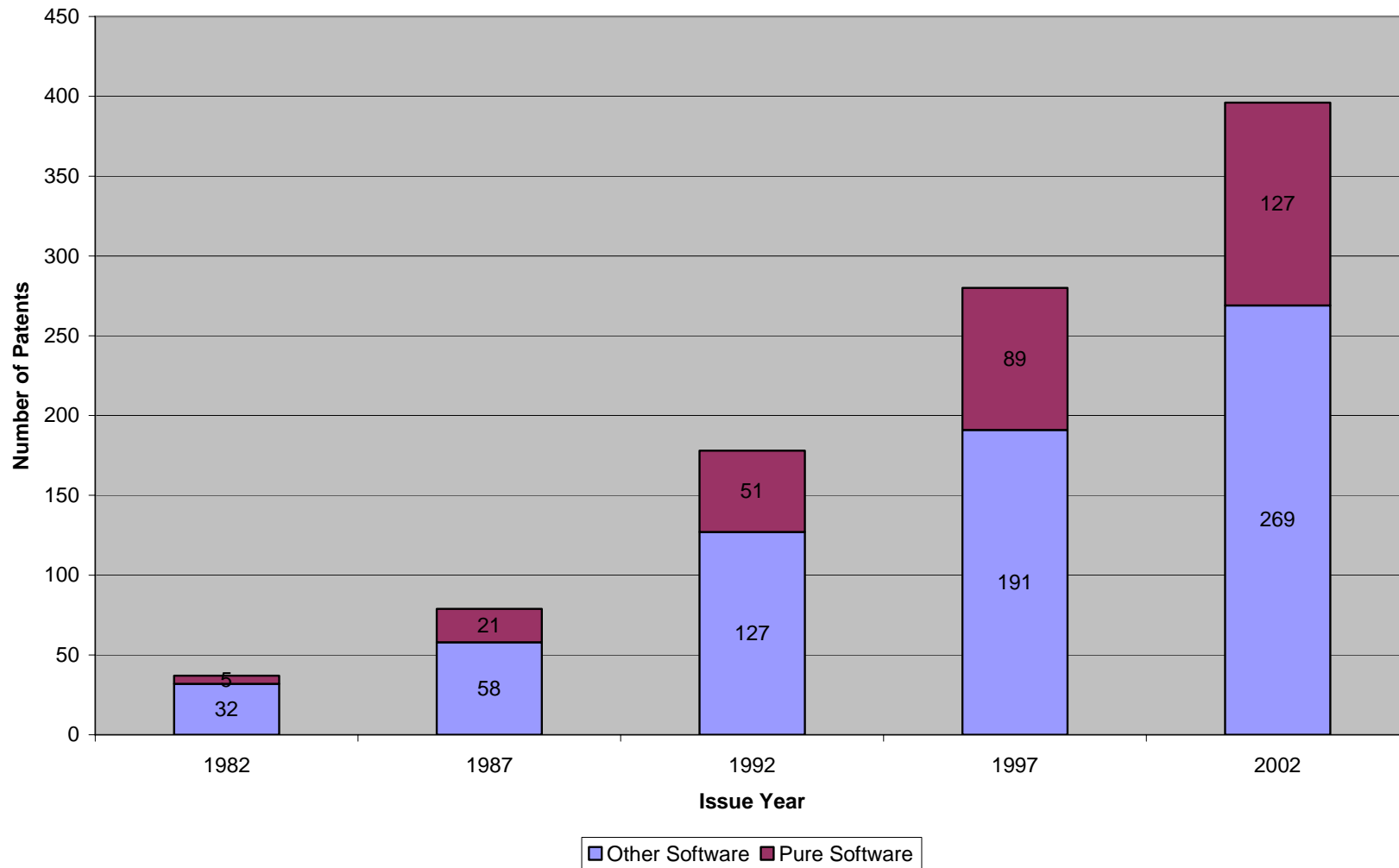
Sampat, Bhaven, 2005. Genome Patents: Bad for Science? (working paper)

Somaya, Deepak, 2005. The Effects of Patent Litigation on University Licensing Efforts (working paper).

Thursby, Jerry and Thursby, Marie, 2006. Where is the New Science in Corporate R&D? *Science*. 314: 1547-1548.

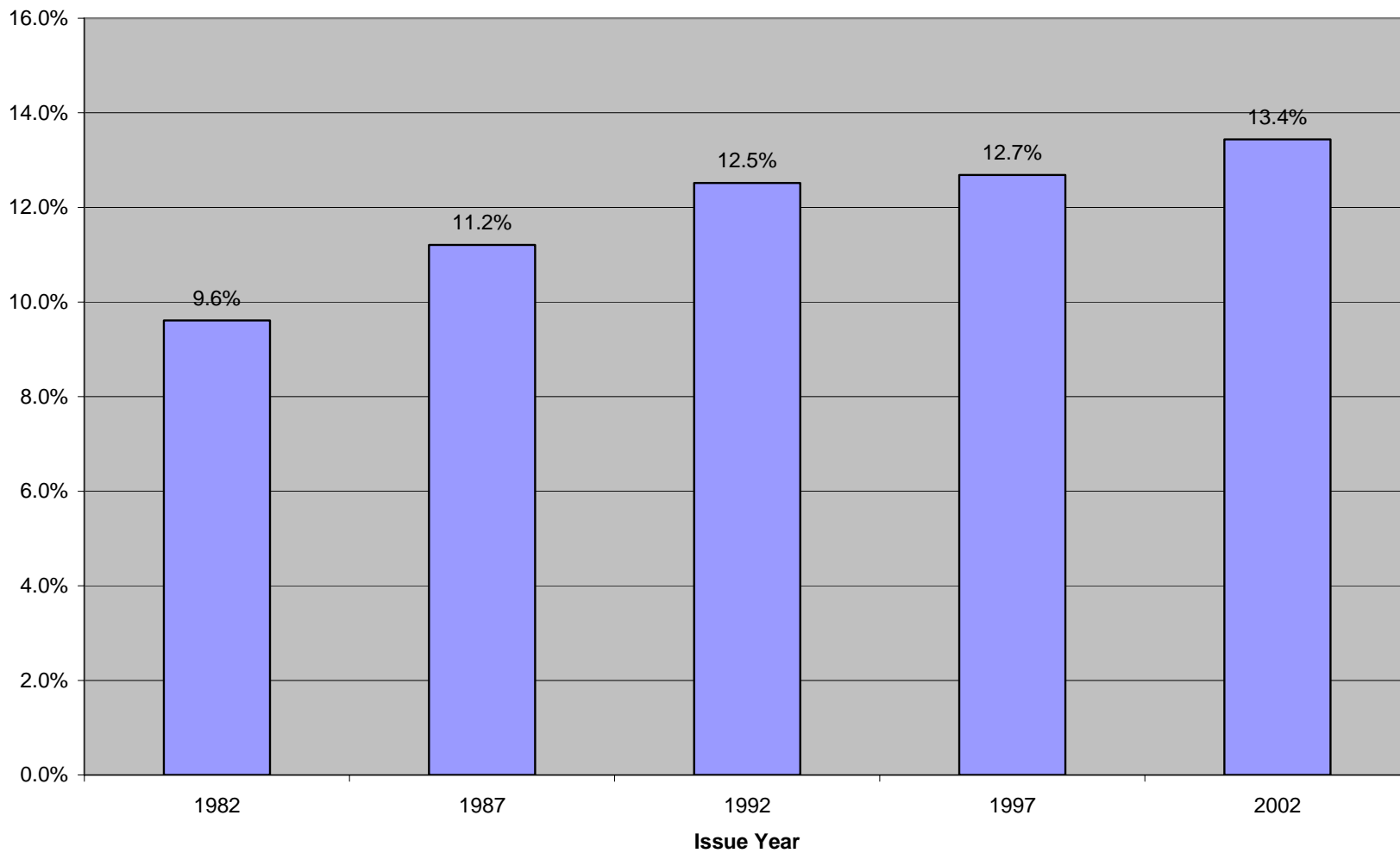
Washburn, Jennifer, 2005. *University, Inc.: The Corporate Corruption of Higher Education*. New York: Basic Books.

**Figure 1: University Software Patents, By Type and Year**



Attachment A – Rai Senate Testimony, Hearing on “The Role of Federally Funded Research in the Patent System”

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



**Table 1: University Software Patenting, Overall Patenting, And Pure Software Patenting In 2002**

<b>University</b>	<b>Rank in Patenting (Issue Year 2002)</b>		
	<b>Software</b>	<b>Overall</b>	<b>Pure Software</b>
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 in the City of New York	15	14	10

Table 2: Negative Binomial Models of Determinants of Software Patents (Issue Year 2002)

	SW (1)	SW (2)	Pure SW (3)	Pure SW (4)
ln(lagged CS R&D)	.110** (.049)	.097** (.045)	.588*** (.121)	.523*** (.110)
ln(lagged Other R&D)	.839*** (.118)	.073 (.141)	.223 (.161)	-.395** (.163)
ln(Non Software Patents)		.910*** (.141)		.781*** (.167)
Const.	-11.577*** (1.406)	-3.613** (1.510)	-9.017*** (1.615)	-2.182 (1.576)
Obs.	202	202	202	202

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

Table 3: Negative Binomial Models of Determinants of Software Patents (Issue Year 1997)

	SW	SW	Pure SW	Pure SW
	(1)	(2)	(3)	(4)
ln(lagged CS R&D)	.245*** (.076)	.172** (.072)	.348*** (.134)	.275** (.131)
ln(lagged Other R&D)	.986*** (.160)	.308 (.196)	1.005*** (.252)	.271 (.320)
ln(Non Software Patents)		.832*** (.185)		.813*** (.282)
Const.	-14.941*** (1.878)	-7.394*** (2.179)	-17.358*** (2.975)	-9.023** (3.587)
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 of Determinants of Software Patents (Issue Year 1992)

	SW	SW	Pure SW	Pure SW
	(1)	(2)	(3)	(4)
ln(lagged CS R&D)	.143* (.077)	.082 (.073)	.122 (.110)	.071 (.107)
ln(lagged Other R&D)	1.083*** (.192)	.348* (.206)	.968*** (.287)	.264 (.323)
ln(Non Software Patents)		1.027*** (.199)		.955*** (.318)
Const.	-15.295*** (2.213)	-7.610*** (2.221)	-14.735*** (3.357)	-7.392** (3.446)
Obs.	202	202	202	202

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

Table 5: Negative Binomial Models of Determinants of Software Patents (Issue Year 1987)

	SW	SW	Pure SW	Pure SW
	(1)	(2)	(3)	(4)
ln(lagged CS R&D)	.343*** (.122)	.314*** (.092)	.289 (.194)	.210 (.162)
ln(lagged Other R&D)	1.013*** (.236)	.291 (.198)	.864** (.351)	.094 (.319)
ln(Non Software Patents)		.797*** (.163)		1.035*** (.303)
Const.	-16.411*** (2.592)	-8.430*** (2.117)	-15.243*** (3.792)	-6.796** (3.261)
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 of Determinants of Software Patents (Issue Year 1982)

	SW	SW	Pure SW	Pure SW
	(1)	(2)	(3)	(4)
ln(lagged CS R&D)	.218*	.167	.770**	.748**
	(.128)	(.124)	(.336)	(.358)
ln(lagged Other R&D)	.612**	-.043	1.435**	.709
	(.270)	(.193)	(.635)	(.774)
ln(Non Software 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  $p < .05$ .

Table 7: Negative Binomial Panel Models of Determinants of Software Patents, Issue Years 1982-2002

	SW	Pure SW
	(1)	(2)
ln(lagged CS R&D)	.037 (.046)	.107 (.096)
ln(lagged Other R&D)	-.344 (.277)	-.119 (.628)
ln(Non Software Patents)	.464*** (.090)	.054 (.167)
1987 Dummy	.665*** (.237)	1.363** (.583)
1992 Dummy	1.354*** (.311)	2.193*** (.754)
1997 Dummy	1.714*** (.379)	2.708*** (.897)
2002 Dummy	2.008*** (.442)	3.045*** (1.040)
Const.	-9.405 (87.124)	-15.118 (418.802)
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  $p < .05$ .