[redacted]
Compsci 82, Group 3

Professor Astrachan
September 17, 2009

The Intellect in Intellectual Property: Absent(minded) in Software Technology?

For my synopsis, I place Mark H. Webbink's article, "A New Paradigm for Intellectual Property Rights in Software," and Richard Stallman's talk, "Software Patents—Obstacles to Software Development" in conversation with one another.[1] I synthesize and discuss the main ideas mutual to the readings: the triviality of software patents, invalid software patents, patents as a barrier to innovation, cross-licensing, and solutions for software patents in a global context. Although Webbink and Stallman both discuss these aspects of intellectual property rights in software, they often approach their concerns differently. Throughout my discussion, I also offer my own reactions to the topics discussed.

Both Webbink and Stallman note that patents are filed at incredible rates for insignificant matters. Webbink approaches this claim by complaining that even the smallest extension of a software technology may be given the same level of protection as a groundbreaking pharmaceutical drug. Stallman does not discuss the slicing and dicing of technology, but instead notes the protection of the trivial by suggesting that even extremely "obvious" ideas receive patents. Webbink and Stallman essentially voice the same concern: the governmental standards for what is significant enough in software to receive protection allows many elements to receive grossly unwarranted patents.

I believe the Amazon one-click patent that Professor Astrachan discussed in class is an example of an idea too obvious to be warranted. Even so, it seems even more ludicrous to me that patent examiners could not at least throw out some of Amazon's claims when

[1] In my reading, I utilized the defined terms provided by the Duke website "Intellectual Property for CS Students" as a meeting point for Webbink and Stallman's respective definitions.

reviewing the broad patent application. Why does the government not insist on narrowing patents down from their expansive, original state?

As a result of trivial patents, there is more than one-patent per one-product and legal monopolies are ultimately created. Consequently, the effort to design even a single new program forces developers to confront a minefield of patents. Thus, Webbink and Stallman claim, the patent system is not a root cause of software innovation; in fact, it retards progress.[2] Initially, I had a difficult time accepting that the same principles of patent law that have positively served the Unites States for so long do *not* promote investment in progress. In fact, going into the articles, I thought that if one was against software patents one would have to be against intellectual property protection in general. Webbink and Stallman, however, have convinced me that patenting software is unique in its curse of removing investment from innovation.

It is not surprising that trivial software patents exist, and that a single product involves many patents, if one considers that there exist hundreds of thousands of software patents. Stallman notes that because of the enormous volume of patents, one cannot keep track of what they are all about; consequently, it is easy to accidentally step on a "mine" and be sued for an idea that has been legally claimed. Moreover, even if one *does* access the long list of patents, the ideas they protect may be hidden by convoluted legal language. Stallman and Webbink are also concerned that because so many patents are produced, individual patents receive little scrutiny by the US Patent and Trademark Office and frequently are invalidated. This point, however, means to *me* that the error in intellectual property rights for software lies not in the patents but in the government offices that control them.

In addition to the *number* of existing patents, the *nature* of each patent may further bar innovation. Webbink notes that firms are turning their backs on research and

[2] This claim, moreover, is buffered by the fact that nations lacking patent protection often see just as much innovation as those *with* patent laws.

development (in other words, on innovation) to embrace patents as a new way to inexpensively expand their patent portfolio. Stallman adds that patents can be broad and basic enough to rule out innovation in an entire field; moreover, innovation can also be ruled out when a company patents a standard protocol. I am not a software designer, and yet I still feel personally slighted by this bar on innovation; it does not merely harm programmers, but also harms software users who are denied choice in the market. I am even more concerned with companies who do not even create a sole standard, but create nothing— companies that amass patents so that they may sue companies who are actually interested in innovation.[3] This, too, limits my choice in the market.

It is possible, however, for a designer to get a hold on a patent via *license*— if the designer is a mega-corporation, that is.[4] Major software companies license patents for income. They also establish cross-licenses among themselves to avoid the troubles and costs of patent litigation. This seems fair to me—after all, isn't the opportunity for wealth and protection the American way? However, I am concerned that individuals and small companies can only rarely engage in such cross-licensing and only rarely can they even afford to purchase a license (and it's a triple whammy that small companies often can't even afford to sue). Stallman makes a crucial conclusion: patents do *not* benefit the small inventor like they do big companies. I wonder: does this imply, then, that maintaining so many patents is evil? If so, who are the evil players in this IP game— the large corporations who apply for the patents, or the legal system that grants them? I also wonder, however, why small inventors even fear patent infringement— aren't they too poor to be sued anyway?

Finally, Webbink and Stallman suggest that there seems to be at least *some* definitive hope in the future of software development— in Europe, that is. The European Patent Office holds that software is not patentable. Both Stallman and Webbink mention terms used by the

[3] These patent collectors can't even be sued because they don't claim to "innovate!"
[4] This is regardless of the fact that small start-ups are most frequently the source for software innovation.

UK Patent and Trademark Office to discuss the European remedy, and here arises (what I believe is) my only confusion among the articles: do the "technical effect" that Webbink mentions and the "technical contribution" that Stallman mentions refer to the same thing? Moreover, I wonder why patent law is one IP law that is particularly inconsistent among so many countries. And is there any way to patent an algorithm at all in a country like the UK?

Despite all of my questions, I have at least made one concrete decision after reading Webbink and Stallman's theories: patents do, in fact, discourage innovation. As for the solution to this dilemma, I am interested in the exclusive use of copyright law as an alternative to the use of software patents. If copyright were exclusively used for software protection, the expression of ideas would be protected while the ideas themselves may not be. Would the consequences of this truly be that detrimental to the software industry? Perhaps software developers should simply return to the old-fashioned way of protecting ideas—keeping them secret for as long as they can. Or perhaps the solution is for the U.S. Patent and Trademark office to begin to, at the very least, hire a vast amount of software designers and learn a thing or two for once about the software they are protecting before they issue their next patent…