From: Martin MacKerel
Sent: Tuesday, April 16, 2013 2:56 AM
To: SoftwareRoundtable2013
Subject: Comments on Software Patents

I have several points to make about software patents, most of which come under the topic of "potential future topics" and many of which, I realize, the USPTO cannot act on without new law from Congress. I'm glad the USPTO is acknowledging that there are some real problems with software patents, but I think that they're asking the wrong questions. They're also asking the wrong people.

To summarize my bottom-line view on software patents, based in part on spending 15 years as a professional programmer and having worked at four startups: software should not be patentable at all. If we allow software patents, there are three things we can do to make them less damaging:

1) reduce the duration to 4 or 5 years instead of 20 (which is a really long time in the software world),
2) increase the bar so only truly innovative things like spreadsheets count, and
3) actually require "reduction to practice" - the submitter must implement it, and release source code, as part of the patent.

Shorten Patent Duration

Twenty years ago the World Wide Web didn't even exist. The pace of change is so rapid that it's ridiculous for software patents to last that long. Five years is more than sufficient. Several IP lawyers were very careful to warn us about not "discriminating" between different types of technology when considering patent rules. While there are some pitfalls, this worry is ridiculous. It serves only to over-protect software patents. Different technologies *are* different. It follows that they should be treated differently. Furthermore, if anything, as other technologies such as 3-D printing and biotech are speeding up in their rate of change, their patent durations should also be shortened. (It was obscene how long the original PCR patent hung over researchers' heads and I understand there remain issues with follow-on patents.) The main pitfall I see to "technology discrimination" is that the lines between software, firmware, and hardware are not strict, and some companies might fudge the line to improve patent chances. Better that we deal with that problem when it arises than continue to deal with the software patent morass we have today.

Raise the Bar

Everyone (except patent trolls) thinks we should raise the bar on patents. What is a significant innovation? What is obviousness? I think that at the very least, software patents should be limited to truly transformative innovations. The invention of the spreadsheet is one. Another candidate might be virtualization. But these are so rare that it seems much better to err on the side of not covering such an innovation than on the side of covering many less important innovations.

I should note that some major inventions definitely benefit from not being patented. The World Wide Web would be a pale shadow of its current self if Tim Berners-Lee or CERN had patented it and attempted to extract royalties. In such a scenario, a technology that has the potential to be cheaply and widely deployed for great public benefit is instead fenced in by a corporation and deployed to a much smaller set of paying customers, in order to make a relatively small gain for itself at a great opportunity cost to the public.

"Source Code or It Didn't Happen"

Many, many patents are just "idea" patents. This mocks the whole rationale behind the patent system. The USPTO should require submitters to have actually implemented the patent. The best way to assure this, and to make it easy for others to duplicate the result, is to require source code. Unlike my suggestion to reduce patent duration, this can be acted on by the PTO without changing any laws.

No, But Really, Abolish Software Patents

It's not that crazy. Software was not always patentable. Many, many programmers, computer science researchers, and economists believe that the government should not grant software patents, and in some cases (e.g. Fed paper on "The Case Against Patents": http://research.stlouisfed.org/wp/2012/2012-035.pdf), economists suggest that **no** patents should be granted in **any** industry.

Impeding Startups

I've worked, as I said, at four startups. In every case, patents were pursued, if at all, later in the game after the technology had been productized and sold, not to protect the market, but as a defensive measure against patent lawsuits from other companies. Everything about the patent filing process was a distraction from the real work - it took time, money, effort, and attention away from the real tasks at hand - improving the technology and selling the product. Application Developers Alliance has an interview with one entrepreneur who has more lawyers than programmers, solely to defend against patent troll lawsuits. The best thing you can do for software startups is to abolish all software patents, immediately.

In some cases, a startup might create a new technology and market and then be overtaken by a bigger rival who copies their innovations. But this happens all the time, anyway. Many good ideas in the software world come up again and again, and in the end it comes down to execution. This benefits the public in that we get the best execution (which is often by the original innovators anyway). Take for example social networking. The idea has been around almost as long as the World Wide Web. Many startups tried and failed to make a business out of it, not because they didn't patent an innovation, but because they failed to execute well. Would we prefer that Six Degrees or Tribe have sputtered on as a PAE, extracting onerous royalties and preventing Facebook from being able to gain the traction that it did?

In addition to being directly threatened by patents, startups are indirectly threatened by patents because of their use of free and open-source software. Every one of the eight or so places I've worked as a programmer has relied heavily on open-source software, for operating systems, languages, web servers, and the like. The free availability of these programs is key to the low cost of entry for new startups. If the Apache web server software or the Python programming language interpreter, for example, were found to be infringing one or more patents, thousands of companies large and small would be panicking to figure out how to respond.

The Public Benefit

The usual story is that patents result in more innovation and more innovation benefits society. But because patents harm startups and smaller companies which are the source of much innovation, the public would benefit from eliminating or weakening patent protection for software. And as I mentioned above, the public benefits from a focus on execution, not priority of discovery.

In addition to indirectly benefiting from open-source software because of its use by startups, the public benefits directly from open-source software in a number of ways. Every major operating system has some components of free software in it (even Apple's and Microsoft's) and major programs like Firefox are open-source. Even non-users benefit by the competition to proprietary software that free software provides.

As an aside, I watched the New York video presentations using the freely available VLC Player. The authors of that program warn that it is "seriously threatened by software patents" because of the "multimedia is a patent minefield". http://www.videolan.org/press/patents.html

The Programmer's Point of View

I want to start with a basic count. Out of 24 speakers at the two roundtables, we have:

14 lawyers
4 law professors
3 entrepreneurs (some of whom are also programmers)
2 programmers
1 investor/incubator

Of the 14 lawyers:

5 were speaking for large companies directly
3 were speaking for organizations that represent the interests of large companies
2 for an organization that represents small software companies/entrepreneurs (Application Developers Alliance)
1 for an IP law firm
1 for a large non-profit research institute that commercializes its inventions
1 for a consumer-advocacy and public-interest-focused non-profit (EFF)
1 for a company very reliant on open-source software (Red Hat)

In other words, out of 24 people, two were non-entrepreneur programmers and one was focused on the public interest at large. Almost everyone else represented the interest of large corporations or entrepreneurs, in other words, capitalists (of different stripes, it is true).

It gets even worse, though. Neither programmer really presented a programmer's point of view on the patent system. One gave a presentation about a kind of broad-view patent analysis that he thought could be helpful in general. The other argued for the use of UML instead of pseudo-code in patents, because it would move the burden from examiners to applicants. This is a small but interesting technical point, but fails to really address the main issues with software patents from the point of view of a working software engineer.

Most programmers are deeply skeptical of software patents. Not all programmers, but many, think that software patents should not exist at all - a point of view that was only brought up in the roundtables to be dismissed. I think this is a serious position that needs to be addressed. Everyone agrees that the current system stifles innovation and has bad consequences for society, and wants to make software patents narrower, harder to get, and easier to challenge. We should give serious consideration to extending this idea to its logic conclusion: that software patents should be impossible to get and cover nothing.

The Class Interest of Programmers

The socio-economic class of programmers is complicated, because they are often paid partly in equity, and at startups this can make them effectively a kind of investor. And, at least in today's environment, the barriers to entry are fairly low for a programmer to become an entrepreneur, sometimes even while holding down a day job.

Nevertheless, I want to consider the bulk of programmers, who are working-class in the broadest definition of the term. That is to say, they must sell their labour to survive. As professionals, programmers (like lawyers) are paid quite well, so while some may chafe at the power dynamics of the capital-labour relation, most are pretty happy about the exchange of labour for money that they get.

The wrinkle that patents throw into the mix is that, by turning a programmer's ideas and techniques into property, he or she can be (and is) dispossessed of it - he or she cannot use the same ideas and techniques later. This impinges on the programmer's future. By contrast, selling one's labour today does not impose obligations on what one may do with tomorrow's labour.

Patents are an incredible example of the *ex nihilo* creation of property. This new property will come (almost by definition) to be dominated by capital, and will therefore increase capital's power *vis-à-vis* its labour force, namely programmers. The large companies that are so well represented at the patent roundtables are giant pools of capital. Their goal is to turn more of the world into property so they can own it and charge rent on it. (The problem, from the point of view of these companies, is that as currently structured the software patent system is interfering with the smooth running of capitalism. I don't think these big companies care about innovation or public interest *per se*; if the technology were completely static and they continued to grow, great. Propertized ideas help them create bigger moats and more advantages of bigness; it actually mitigates the threat from disruptive new entrants to the market. But the threat of patent trolls has grown so large that something needs to be done. But not too much. Patents have a high cost to defend, which leads to economies of scale and favours larger companies.)

In addition to the basic economic view, many programmers dislike the very idea that some legal fiction would prevent them from creating software. Programming is, among other things, an art and a craft, and such restrictions raise programmers' hackles. And there is a solidarity, too, that programmers don't like helping to restrict other programmers in this way. Many companies will promise that they will only use patents defensively, and they may mean this, but their word alone can't guarantee it. Companies may change ownership, or go bankrupt and have their patents sold off. Or they may simply change their minds, as Yahoo did - see "A Patent Lie: How Yahoo Weaponized My Work" - http://www.wired.com/business/2012/03/opinion-baio-yahoo-patent-lie/ . Twitter at least has made some progress in finding a way to make a legally meaningful promise.

Leaving Options Open for a Non-Capitalist Economy

If I haven't lost you by explicitly talking about class, labour, and capital, consider this more radical suggestion. As of this writing (April 2013), capitalism has been in a serious worldwide crisis for more than five years. Even if it were functioning well on its own terms, we face multiple environmental disasters, most notably the continued increase of carbon dioxide and other greenhouse gases in our atmosphere, which unchecked will lead to truly catastrophic climate change. Capitalism's quest for constant growth means that we are producing **more** every year, when we need to be producing **less**, and consuming less energy.

Obviously replacing capitalism is a tall order, and not something I'm suggesting the USPTO attempt here - it exists to bolster capitalism. But we should think about how we, as a society, can keep our options open for the future. A rich ecosystem of free and open-source software (FOSS) is a good start to a non-capitalist economy, and so we should protect and encourage the development and adoption of FOSS. Software patents are a grave danger to FOSS and for the societal resilience that such software gives us.

Martin MacKerel