The American

Software Patents: Reform, Not Repeal

By Michael M. Rosen Friday, December 9, 2011

As we seek to further calibrate the delicate balance so critical to our regime of incentivizing innovation, we should reform software patents, not repeal them.

Another day, another call for fundamental change to our patent system.

As I explained in this space a few months ago on the heels of a widely circulated NPR report on so-called “patent trolls” and news of software giants like Microsoft and Google arming up with patents in anticipation of an apocalyptic intellectual property war, Congress at long last passed the most significant overhaul to our patent system in decades.

But for some patent skeptics, like Cato Institute scholar Timothy B. Lee, these changes don’t go far enough, especially in the area of patents on software.

In a National Review article entitled “Patently Absurd,” Lee frontally assaults software patents, claiming that they stymie innovation. The fundamental problem, Lee asserts, is the following:

> While a small team of brilliant engineers can build some of the world's best software, it has no hope of keeping up with big companies' rate of patent filings. Patents threaten to turn Silicon Valley into a place where new firms must develop large legal bureaucracies before they can challenge incumbent firms.

Lee contends that “the explosive growth of patent trolling is just one of the many problems created by our dysfunctional patent system.” He focuses on one particular such troll—a Texas outfit called DataTreasury, which Lee describes as having “launched a patent-litigation campaign against the nation’s banks” and having collected “hundreds of millions of dollars in licensing fees and damages.” (Full disclosure: I have worked on matters involving DataTreasury and will therefore refrain from commenting about the company.)

There are three principal reasons, Lee contends, that software patents are a bad idea: “First, software development is an individual, creative activity, more akin to writing a novel than designing a jet engine. A single programmer can inadvertently infringe dozens of software patents in the course of a single project.”

Second, he asserts that “software patents are especially prone to litigation,” citing as evidence a study showing that patent litigation costs began to skyrocket in the 1990s.

Third, Lee posits, “software patents are unnecessary because software is already eligible for copyright protection.”

But ultimately, Lee’s criticisms of software patents—while elegantly presented and sincere—fall short of the mark. All of the software patent problems that Lee raises need not result in the extermination of the software patent species, but instead can be remedied simply by improving the Patent Office processes and standards by which software patents are examined, granted, and reviewed.

First, superficially, Lee’s reflections are more than a bit self-contradictory: on the one hand, he faults patent trolls for exploiting the patent litigation regime by shaking down large, wealthy software companies. But he also blasts those large software companies for obtaining so many patents that they supposedly screw the little guy. So who is it, exactly, that’s manipulating the system? Everyone? Actually, by tightening standards for awarding software patents—rather than eliminating them altogether—the trolls will have less incentive to attack industry behemoths, while those behemoths, in turn, will have less reason to arm up in self-defense.

Second, Lee’s explanation of why software patents do more harm than good omits certain key considerations.

Any Tom, Dick, or Harry can develop the next Angry Birds, and we should encourage such innovation by ensuring that developers can protect their inventions.

With respect to the personalized nature of software, precisely because “software development is an individual, creative activity,” we should take great pains to protect the inventive effort that flows from such activity. As the emergence in recent years of an entirely new market in mobile computing applications has shown, there are virtually *no* barriers to entry in the software field. Any Tom, Dick, or Harry can develop the next Angry Birds, and we should encourage such innovation by ensuring that developers can protect their inventions.

With respect to the likelihood of software patent-related infringement lawsuits, we shouldn’t be surprised that a high proportion of patent litigation pertains to software, since so much of our lives now revolve around computing. In addition, a key driver of the recent increase in patent litigation costs has been the migration of patent cases from the bench to the jury box. According to a PriceWaterhouseCoopers study, in the 1990s, three patent cases were tried to a judge for every one case tried to a jury; by the 2000s, the ratio had slipped to one to one. For reasons requiring an entire article of their own, jury trials are far more expensive than bench trials, especially in the patent field.

Finally, the new patent reform legislation helps restore balance to the system, including in the software patent realm. For his part, Lee faults the recent bill as contributing to the problem, not the solution. “The America Invents Act is full of such technocratic provisions,” Lee writes, “that tilt the playing field toward big businesses without doing anything to address the system's deeper flaws.”

Moreover, while copyright law can protect individual lines of software code, it affords no safeguard to the inventive ideas underlying that code.
This casual analysis, however, overlooks several important provisions of the new act—including the establishment of a new post-grant review to knock out bad patents early on; an expansion of what qualifies as a prior product that can render a patent invalid; tightening up standards for “willful” infringement, which can result in treble damages; and bolstering the Patent Office’s authority over its funding. Those provisions will rein in some of the excesses of the system, including (especially?) among software patents.

Coupled with recent changes in the case law that have curtailed massive damages awards, made it easier to invalidate patents as obvious, helped defendants escape patentee-friendly jurisdictions, and rendered it more difficult for patentees to obtain injunctions, this legislative overhaul goes a long way to curing problems in the patent system.

So, as we seek to further calibrate the delicate balance so critical to our regime of incentivizing innovation, we should heed the criticisms of folks like Lee. But we should also take other key evidence into account. And in the end, reforming software patents, not repealing them, will prove the most prudent course.