Giving the Software Field Protection from Patents

by **Richard Stallman**

A version of this article was first published at Wired in November 2012.

Patents threaten every software developer, and the patent wars we have long feared have broken out. Software developers and software users—which in our society, is most people—need software to be free of patents.

The patents that threaten us are often called “software patents”, but that term is misleading. Such patents are not about any specific program. Rather, each patent describes some practical idea, and says that anyone carrying out the idea can be sued. So it is clearer to call them “computational idea patents”.

The US patent system doesn't label patents to say this one's a “software patent” and that one isn't. Software developers are the ones who make a distinction between the patents that threaten us—those that cover ideas that can be implemented in software—and the rest. For example, if the patented idea is the shape of a physical structure or a chemical reaction, no program can implement that idea; that patent doesn't threaten the software field. But if the idea that's patented is a computation, that patent's barrel points at software developers and users.

This is not to say that computational idea patents prohibit only software. These ideas can also be implemented in hardware—and many of them have been. Each patent typically covers both hardware *and* software implementations of the idea.

The Special Problem of Software

Still, software is where computational idea patents cause a special problem. In software, it's easy to implement thousands of ideas together in one program. If 10% are patented, that means hundreds of patents threaten it.

When Dan Ravicher of the Public Patent Foundation studied one large program (Linux, which is the kernel of the GNU/Linux operating system) in 2004, he found 283 US patents that appeared to cover computing ideas implemented in the source code of that program. That same year, a magazine estimated that Linux was .25% of the whole GNU/Linux system. Multiplying 300 by 400 we get the order-of-magnitude estimate that the system as a whole was *threatened by around 100,000 patents*.

If half of those patents were eliminated as “bad quality”—i.e., mistakes of the patent system that is—it would not really change things. Whether 100,000 patents or 50,000, it's the same disaster. This is why it's a mistake to limit our criticism of software patents to just “patent trolls” or “bad quality” patents. The worst patent aggressor today is Apple, which isn't a “troll” by the usual definition; I don't know whether Apple's patents are “good quality”, but the better the patent's “quality” the more dangerous its threat.

We need to fix the whole problem, not just part.

The usual suggestions for correcting this problem legislatively involve changing the criteria for granting patents—for instance, to ban issuance of patents on computational practices and systems to perform them. This approach has two drawbacks.

First, patent lawyers are clever at reformulating patents to fit whatever rules may apply; they transform any attempt at limiting the substance of patents into a requirement of mere form. For instance, many US computational idea patents describe a system including an arithmetic unit, an instruction sequencer, a memory, plus controls to carry out a

particular computation. This is a peculiar way of describing a computer running a program that does a certain computation; it was designed to make the patent application satisfy criteria that the US patent system was believed for a time to require.

Second, the US already has many thousands of computational idea patents, and changing the criteria to prevent issuing more would not get rid of the existing ones. We would have to wait almost 20 years for the problem to be entirely corrected through the expiration of these patents. We could envision legislating the abolition of these existing patents, but that is probably unconstitutional. (The Supreme Court has perversely insisted that Congress can extend private privileges at the expense of the public's rights but that it can't go in the other direction.)

A Different Approach: Limit Effect, Not Patentability

My suggestion is to change the *effect* of patents. We should legislate that developing, distributing, or running a program on generally used computing hardware does not constitute patent infringement. This approach has several advantages:

- It does not require classifying patents or patent applications as “software” or “not software”.
- It provides developers and users with protection from both existing and potential future computational idea patents.
- Patent lawyers cannot defeat the intended effect by writing applications differently.

This approach doesn't entirely invalidate existing computational idea patents, because they would continue to apply to implementations using special-purpose hardware. This is an advantage because it eliminates an argument against the legal validity of the plan. The US passed a law some years ago shielding surgeons from patent lawsuits, so that even if surgical procedures are patented, surgeons are safe. That provides a precedent for this solution.

Software developers and software users need protection from patents. This is the only legislative solution that would provide full protection for all. We could then go back to competing or cooperating… without the fear that some stranger will wipe away our work.

See also: Patent Reform Is Not Enough