What’s wrong with software patents?

May 17, 2008 · Filed under Software patents & innovation

[This post is the third in a series of posts based on empirical research in a new book, *Patent Failure*, by James Bessen and Michael Meurer.]

Patents on computer programs, financial processes and business methods have been controversial at least since the 1960s. Surveys regularly find that computer programmers are opposed to patents on software by a wide margin. In what other field is the class of inventors so opposed to patents?

Is there a problem?
Some people contend that there is nothing particularly wrong with patents on software, arguing that “patent thickets” are not preventing innovators from entering software markets. But the latest evidence suggests that patent thickets do, in fact, inhibit startups in software. More important, patent thickets might not be the only or even the most important problem with software patents.

Indeed, our evidence suggests an important problem of another sort: software patents are four times more likely to be litigated than are chemical patents; business methods patents are twelve times more likely to be litigated; finance patents are 49 times more likely. Moreover, the evidence also suggests that these patents have lower values than chemical patents, so these patents are not being litigated more because they are more valuable.

Other people admit that there are problems with software patents, but they suggest that this is only temporary: once judges and patent examiners understand this technology better, once they have become familiar with the prior art, etc., then the uncertainty about these patents will abate and litigation rates will go down. But the evidence shows that after a decade of issuing software patents in large numbers (over 200,000), the probability that a newly-issued software patent will be litigated is continuing to rise.

So it does seem that patents on software and related technologies at least have a particular problem with litigiousness. And this problem is central to the poor performance of the patent system generally. In the previous post we highlighted how litigation costs have substantially outgrown the profits that public firms receive from patents outside of the pharmaceutical and chemical industries. In 1999, 38% of the cost of litigation among public firms arose from lawsuits involving software patents; preliminary data suggest that this share has increased since then. Litigation over software patents is clearly a major factor in the poor performance of the patent system. So in a very real way, the overall performance of the patent system cannot be fixed unless the particular problems of software patents are also fixed.

Why are there so many lawsuits over software patents?

A variety of evidence leads us to conclude that software patents are involved in relatively more litigation because they are more likely to have "fuzzy boundaries." Statistical evidence shows, for example, that software and business method patents are much more likely than other patents to have their claim construction appealed to the Federal Circuit. Part of this tendency arises because of the nature of the technology and part arises because of the way the courts have treated this technology.

Our reading of the case law convinces us that patent law tolerates too many software claims untethered to any real invention or structure; in such a world clear boundaries are unattainable. When patent claims relate to actual devices or chemical structures, then their meanings can be interpreted by reference to those physical or chemical entities. However, when the words refer to abstract ideas, they are often subject to multiple interpretations and are therefore more ambiguous. For example, many people thought that "point of sale location" (in the famous E-Data patent) was computer industry jargon for that place in a retail store where transactions take place, formerly occupied by a cash register. When the Federal Circuit interpreted this claim, they decided that it referred to any location where an e-commerce transaction might take place, although it is highly unlikely that this is what the inventor had contemplated 17 years earlier. Thus a broadly worded invention for a kiosk in a retail store was read to cover a broad swath of e-commerce. Not surprisingly, this patent generated quite a few lawsuits.

Patent doctrines that might serve to prevent such fuzzy claims have been undermined. For example, the enablement doctrine has historically been used to keep patents from claiming much more than what was actually invented. Unfortunately, as a result of Federal Circuit decisions on software patents during the 1990s, these patents no longer need to provide computer code, a flowchart, nor any detailed description of specific operation in order to be enabled.

Fixing the problem

A lot of people have very strong opinions about how patent policy should or should not change regarding software patents. We wish we had such clarity, but we do not. We are convinced that the current treatment of software patents creates significant problems and that these are getting worse. But the problem is complex and fixing it will likely involve multiple changes in law and institutions.

Certainly, *KSR*, the recent decision on obviousness, should help and so would a stronger indefiniteness requirement. Additionally, it might help to restore a substantial enablement requirement for software patents so that these patents are restricted to claiming more or less what was actually invented and disclosed.

Possibly, a subject matter test might help. We confess we do not have a rule that cleanly distinguishes inventions using software that should be patentable from abstract processes that should not be patentable. Some people argue that any attempt to proscribe subject matter will only increase uncertainty and encourage avoidance through clever claim drafting. But the evidence suggests that the subject matter tests used following *Benson* and *Flook* did not, in fact, encourage excessive litigation during the 1980s, even though there was some evasive claim drafting. Litigation rates for software patents then were about the same as those for all patents.

On the other hand, we doubt that a subject matter test by itself would be sufficient to fix the problems of software patents.