The Return of Functional Claiming

Mark A. Lemley
William H. Neukom Professor, Stanford Law School
Partner, Durie Tangri LLP
mlemley@law.stanford.edu

Patent law purports to promote innovation by giving inventors the exclusive right to their inventions. In fact, however, modern patent law pays far less attention to what the patentee actually invented than to the patent "claims" – the legal definition of the scope of the patent drafted by lawyers. And lawyers have a natural tendency to broaden those claims as much as possible in order to secure the strongest possible rights for their clients. The result, particularly in the software and Internet industries, has been a proliferation of patents with extremely broad claims, purporting to own everything from international electronic commerce to video on demand to emoticons to means of hedging commodity risk.

Patent law has faced this problem before. Seventy-five years ago, in the wake of the law's move away from a focus on what the patentee actually built towards what the lawyers defined as the boundaries of the invention, patent lawyers were increasingly writing patent claims in broad functional terms. Put another way, patentees were claiming to own rights not a particular machine, or even to a particular series of steps for achieving a goal, but to the goal itself. The Supreme Court ultimately rejected such broad functional claiming in the 1940s as inconsistent with the purposes of the patent statute. When Congress rewrote the Patent Act in 1952, it adopted a compromise position: patentees could write their claim language in functional terms, but when they did so the patent would *not* cover the goal itself, but only the particular means of implementing that goal described by the patentee and equivalents thereof. These "means-plus-function" claims permitted the patentee to use functional language to describe an element of their invention, but did not permit her to own the function itself however implemented.

Functional claiming is back. While experienced patent lawyers generally avoid writing their patent claims in means-plus-function format, software patentees have increasingly been claiming to own the function of their program itself, not merely the particular way they achieved that goal. Both because of the nature of computer programming and because of the way the means-plus-function claim rules have been interpreted by the Federal Circuit, those patentees have been able to write those broad functional claims without being subject to the limitations of section 112(f).

Commentators have observed for years that patents do less good and cause more harm in the software industry than in other industries such as pharmaceuticals. Software patents create "thickets" of overlapping inventions, and are asserted in droves by patent "trolls" against innovative companies. Some have argued that software isn't the sort of thing that should qualify as an invention at all. Others have pointed to the laxity of the Patent and Trademark Office (PTO), which they say has allowed too many

patents on obvious software inventions. Still others say that the problem is the absence of clear boundaries, so that it is impossible to know whether a patent claim covers a particular product without going to court to get a ruling on what the patent means.

While there is some truth to each of these criticisms, it is broad functional claiming of software inventions that is arguably responsible for most of the well-recognized problems with software patents. Writing software can surely be an inventive act, and not all new programs or programming techniques are obvious to outside observers. So even if there are too many software patents, the patent thicket and patent troll problems won't go away if we simply reduce the number of software patents somewhat. And while the lack of clear boundaries is a very real problem, the most important problem a product-making software company faces today is not suits over claims with unclear boundaries but suits over claims that purport to cover any possible way of achieving a goal. The fact that there are lots of patents with broad claims purporting to cover those goals creates a patent thicket. And while the breadth of those claims should (and does) make them easier to invalidate, the legal deck is stacked against companies who seek to invalidate overbroad patent claims.

While there are some benefits to broad functional claims in software, they are insufficient to justify the costs they impose. As it did seventy-five years ago, the law should rein in efforts to claim to own a goal itself rather than a particular means of achieving that goal. Doing so should not require legislative action; it is enough to interpret existing section 112(f) in light of the realities of software and modern patent practice.

In Part I, I discuss the history of functional claiming and how it was cabined. In Part II, I describe the explosion of functional claims in software and how they have managed to skirt the limits imposed on functional claiming. In Part III, I argue that functional claiming in software is responsible for many of the ills that beset the software patent system. Finally, in Part IV, I argue that the problem could be solved simply by applying the rules of means-plus-function claims to software. While doing so would narrow the scope of software patents, unfairly in a few cases, on balance the social benefits would be substantial.