March 12, 2013
To: United States Patent & Trademark Office
From: Michael Risch, Villanova University School of Law[1]
risch@law.villanova.edu

Re: Request for Comments on Functional Claiming and Software Patents
Docket No. PTO-P-2012-0052

I was sorry to miss the roundtables on this topic. I respectfully submit my comments in response to the above request for comments.

These comments question the ability of 35 USC § 112(f) to solve the problem of software patents, assuming we can agree on what that problem is. That said, I agree that indefiniteness through § 112(f) is one tool to invalidate software patents when appropriate. I also applaud any effort to shed light on functional claiming, which is problematic for many reasons I discuss below. However, my concern is that use of § 112(f) will not necessarily make software more certain, nor do I believe that the primary problem with functional claiming is even lack of certainty. Instead, functional claiming implicates each of novelty, nonobviousness, description, and enablement. The remainder of these comments explains how.

Functional Claiming, Alappat, and Novelty

The rise of functional claiming – at least widespread acceptance of it – can be traced to In re Alappat, and its legal fiction that old hardware becomes new again if it is configured with different functionality. We live with this fiction because it allows for patenting of machines rather than methods, but functional claiming is the necessary by-product. The easiest way to solve the functional claiming problem is to reverse the rule of Alappat, and recognize reality: machines do not become new simply because new software is loaded onto them. Algorithms are not structure. Trying to fit algorithms into the framework of structure is problematic in that sense alone, because, as discussed below, we have little way of knowing when an algorithm is sufficiently defined as to be "structural," nor do we know what its equivalents are.

Methods are Algorithms

Even if Alappat were reversed, patentees would not be without protection for inventions, of course. They could claim methods that perform the series of steps on a computer. Such claims would be a new use of the machine under 35 U.S.C. § 100(b). Indeed, one of my primary concerns with reliance on § 112(f) is that the very same functions would be valid if claimed as a series of steps. In other words, the functional capabilities *are* the algorithm. The "structure" is merely incidental in such claims, and that's the whole point of method claims in the first place.

This means that we allow things in method claims (and in non-means-plus-function apparatus) claims that we don't allow under § 112(f). That's OK, but we should be explicit about where § 112(f) might apply, and where it might not. I believe that this is a real shortfall in the suggested approach.

[1] University affiliation is provided for identification only. These comments reflect the views of the author and no others, including Villanova University.

This shortfall is compounded by a lack of understanding about what equivalents might apply to an algorithm. The proposal implies that the § 112(f) would result in narrower claims than a comparable claim that includes "structure" in the claim. But this cannot be right if we take the history and meaning of §112(f) seriously. One of the points of the statute was to ensure that patentees would *not* be limited to the exact structure that they describe, but really any structure that is equivalent and performs the same function. So, § 112 (f) gives with one and takes away with the other. It might invalidate a few patents that have *no* algorithm, but it might not limit the scope of patents to their particular algorithms—it might expand them! This expansion can be contained if equivalents are taken seriously and prescribed narrowly, but that is supposed to happen already, and it seems to be failing.

Written Description

Mark Lemley's article notes that a primary problem of functional claiming is that it allows for broad patents covering far more than what the inventor invented and described. The goal, it is clear, should be to limit to inventor to what was invented, rather than what was claimed. This goal, while important—indeed, critical!, is not about indefiniteness. It is about written description. In other words, in many cases, the concern about functional claiming is not that the claim is indefinite. In most cases, one knows exactly the "means" that is performing the function. The problem is that the claimed function is far too broad for the disclosure.

But this problem is best managed by asking what the inventor invented, and determining whether the inventor conceived of a claim as broad as that included in the patent. This may be preferable to § 112(f) because it allows the PTO and courts to focus on the entire claim, and not piecemeal elements. It may also be preferable, because equivalents need not be considered; instead, the question is whether the inventor actually invented the thing at issue.

Obviousness

Another restatement of the broad patent coverage is that functional language allows patentees to own the solution, rather than the implementation. But this, too, is not really an indefiniteness problem under § 112(f). To be sure, in many cases the inventor will fail to disclose how they actually solved a problem, but if the inventor discloses the algorithm used, then he or she will *still* be able to claim the broad solution, where the solution *is* the algorithm. That concern is not indefiniteness, but obviousness.

To date, and in other areas of patenting, we have given credit to inventors that have conceived of (nominally) non-obvious solutions that were obvious to implement, even if the obvious implementation locked out the field. The question is whether such an approach should continue for software patenting. That is beyond the scope of these comments, but I want to point out that obviousness is a core problem with most software patents—at least the hundreds on which I have conducted prior art searches. The patentee has solved some problem (one-click ordering, reverse auctions, shadow escrow accounts) that is relatively easy to implement. But once they have solved the problem, others cannot perform the same method, and § 112(f) will not make life easier. Patentees may well disclose the algorithms in their patents. Further, they

may disclose them in method claims. And the patent continues, despite appearing to be relatively obvious and easy to implement.

Enablement

Also unclear in the proposal is how § 112(f) might be implemented. As Colleen Chien's roundtable presentation shows, many hot-button patents – including extremely broad ones – have detailed disclosures with detailed algorithms. At the very least, such patents disclose a "functional abstraction" of "what the computer will do." This is a form of algorithm, and thus satisfies the "structure" requirement as proposed and currently applied. Many patents appeared to do better than functional abstractions, and included pseudocode and even data structures.

Of course, functional abstraction is not the ideal way to present an algorithm. If the algorithm is presented as a series of steps the computer will perform, then the PHOSITA might need to do a bit of work to determine just what code and data structures are necessary.

But this, too, is only a question of definiteness in the marginal case. If the issue is "what the program will do" with respect to the entire claim, then it is surely indefinite. For example, if the specification merely stated that, "the program will present a reverse auction," then it might be indefinite.

However, if the functional abstraction is present for each means + function step, then the PHOSITA would know the algorithm. For example, if claim includes "means for a user to input a number," and the specification includes the functional abstraction, "the program will present a data field that accepts input from the user," then the algorithm would be satisfied.

Or would it? This is unclear from the proposals. But the way we would decide that question is no longer one of definiteness, but of enablement. We ask whether the PHOSITA can read and understand the invention. This comes as no surprise. When we worry about breadth of an invention in light of a disclosure, enablement is usually the first doctrine that comes to mind. I submit that it must be considered as part of the solution.

In summary, I agree that functional claiming is a problem. I also agree that failure to disclose structure in means-plus-function claiming is a problem. I am, unfortunately, not convinced that application—even stricter application—of § 112(f) is a panacea. I wish there were some panacea, but problems with overbroad and unclear software patents go much deeper, to novelty, obviousness, description, and enablement, among other things.

Respectfully submitted,
/s/
Michael Risch