From: Shawn Rutledge [e-mail redacted]
Sent: Monday, September 27, 2010 4:05 PM
To: Bilski_Guidance
Cc: [e-mail redacted]
Subject: comments regarding new guidance for patent applications

I'm writing because of the request for comments regarding guidance about which patent applications are considered acceptable. I'm a software engineer, and have always thought that software patents are a quagmire, a restriction of freedom with no benefit.

Software patents threaten to take away our ability to control the devices that now exert such strong influence on our personal freedoms, including how we interact with each other. Now that computers are near-ubiquitous, it's easier than ever for an individual to create or modify software to perform the specific tasks they want done -- and more important than ever that they be able to do so. But a single software patent can put up an insurmountable, and unjustifiable, legal hurdle for many would-be developers to freely distribute whatever modifications they may invent. For such developers, often there may not be any profit motive... it's simply a matter of making improvements which are useful to people, or fixing things which are broken or which hinder the usefulness of a device.

In addition the very fact that software patents exist becomes an extra burden for companies. They are like blunt force weapons in that large companies focus on accumulating a portfolio of patents that they can use as the basis for lawsuits against competitors. The lawsuits in many cases are frivolous, because the USPTO has not set the bar high enough for patentability, so it is possible for companies to patent a lot of minor developments which do not significantly advance the state of the art. Then they use those patents as a shield: if you sue us we can sue you too. If you compete too much maybe we will sue anyway, just because we can, and we can use the money. Or heck, come to think of it we can use the money anyway, so let's just sue everybody. Consequently this shield has a chilling effect on startups which do not have a counter-portfolio of silly patents that they can use for counter-suing, and even more of a chilling effect on free software developers who have no profit motive, but just want to use the "invention" anyway (which in some cases is so obvious that it can be re-invented by anyone with some experience). And any company or individual who participates is wasting energy and manpower and resources just on the legal process rather than in actually doing something useful. So this is the sense in which I argue that the very existence of software patents becomes a burden borne by everyone who wishes to participate in software engineering.

Then there is the matter of the 20 year maximum validity. That is way too long, but at least we are ensured that 20 years after an invention, it will be able to be used freely. This pattern has actually occurred often enough: for example because of the zip algorithm patent, and because Unisys (which had no other significant revenue anymore, having become an ailing dinosaur by that point in time) had become a patent troll, we saw a transition away from .gif images on the web. There was nothing wrong with GIF, and the replacement PNG format is nevertheless superior in some ways, but a lot of manpower was squandered in the mad rush to replace all of the

GIF images with PNGs before Unisys got around to suing in any particular instance where the suit was possible. Now, after all of that, it's finally moot because thank goodness that stupid patent has finally expired. 20 years after the invention we are finally free to use GIF (and hardly anybody cares because PNG is superior anyway). The speed of software innovation is such that 20 years is much worse than, say, 5 years - this obstacle of not being able to use an invention (which in some cases can hardly even be called an "invention" at all) will be there from cradle to grave, because in 20 years it's no longer relevant. But I would argue further that even 1 year is too long, because on the whole, software patents are a drag on innovation.

The job of the patent office was intended to encourage innovation by 1) encouraging the publishing of inventions, as opposed to keeping them as trade secrets 2) giving the inventor a temporary short-term monopoly as a reward for the publishing, to help the inventor to entrench his profits before the competition catches up. But instead what we see is that publishing is not a problem anymore with the internet, and with the free software movement - most of the interesting new ideas get published anyway, as long as the law allows. The existence of patents becomes an obstacle to publishing further instantiations of new ideas, beyond the patent itself. If there was no patent, the new idea would be used over and over again in free software which by its very nature is being published. As for the monopoly, 20 years is way too long as I have mentioned above - often the invention will be obsolete before it can be used.

Another problem with patents in general is that licensing is not mandatory. If an inventor wants to patent something and then refuse to license it, he's free to do so. He can even subsist merely on lawsuits if he chooses. He can block progress completely on that front, and still line his own pockets, while clogging up the legal system too. This antisocial behavior is a drag on innovation, and really needs to be stopped - in every field ideally, but in the "soft" fields most of all, where progress is so much faster than it was during the industrial revolution.

The Supreme Court of the United States has never ruled in favor of the patentability of software. Their decision in Bilski v. Kappos further demonstrates that they expect the boundaries of patent eligibility to be drawn more narrowly than they commonly were at the case's outset. The primary point of the decision is that the machine-or-transformation test should not be the sole test for drawing those boundaries. The USPTO can, and should, exclude software from patent eligibility on other legal grounds: because software consists only of mathematics, which is not patentable, and the combination of such software with a general-purpose computer is obvious.