From: Jeffrey Yang [e-mail redacted]
Sent: Saturday, September 25, 2010 11:41 PM
To: Bilski_Guidance
Subject: Software as an invention

I am writing to describe my beliefs on the value of software inventions.

I'll quickly describe my basic assumptions:
1) Software is a description of a process for handling data.
2) People who write software must save their software in the form of files, i.e. documents.
3) When used in a certain fashion, software will provide value for the user (saving time, performing a service for which the user does not have the skill, etc.)

While I will skip some more assumptions and observations, I'll share my conclusion that software is only patentable when it can be tested and statistically proven that
A) the new process (relating to assumption 1) is creative
B) the person/legal entity wrote their own implementation of the process
C) the software has been shown to provide, ex post facto, business value

To be more specific, I am suggesting that the USPTO implement bright line policies and testing procedures for any software, such that the patent office does not grant patents unless it can provide a report, to the public in general, that shows the software has passed certain thresholds.

How do you implement such testing procedures? In reference to the above, in reverse order:
C: Has the software been released to the public (or a minimum threshold of users)? No? Then it cannot be patentable. This is the easiest test, and must be enforced to ensure that vaporware and other currently-not-implementable ideas are not awarded patents. This is an incentive (Freakonomics-style) for businesses to actually apply for patents based on man-hours and capital expenditures, versus packaging ideas in software languages and passing it off as a functioning device.

B: Does the person/legal entity have the original source code? It should be submitted to the USPTO, securely, and run through a source code analysis package. This would be stored as a digital fingerprint of the software, and as long as it doesn't match any other previously awarded fingerprint, the software could be considered unique. So, it's ok if you and I both have the exact same idea, if mine is written for a smartphone browser and yours is written for the Playstation 3 web browser, then our software fingerprints will likely be very different, and we will likely be awarded distinct patents (assuming other criteria are met). However - if somehow our source code was very similar (because we both follow standard practices and patterns, and used similar development tools), then at most 1 patent will be awarded (because only one of us had the idea and implementation first), and test A would take precedence in determining who, if anyone, would get a patent.

A: Have a panel of about 30 software developers try to implement the exact same process, given the facts and state of the art relating to the case. If a statistically significant number would have described and designed that same process, it is an obvious work of art and not allowable.

A couple other remarks:
Bright lines policies often are "gamed" by applicants who know the rules or have the resources to do so. This is irrelevant to the value of the policies, because as long as the policies are easily understood, they will easier to adjust. The fact that source code will be submitted is another deterrent to junk software patents, because the resources of the USPTO are limited, and the risk and onus lies on the applicant to provide well-defined, information-rich patent applications, which will be processed by an automated analysis machine at the USPTO.
The panel of reviewers is an incentive for both the USPTO and the applicant to innovate. The USPTO must find other ways to study the very subjective parts of a patent application - I would not be opposed to actually allowing/enabling/forcing the applicant to share the idea (not the implementation) to the general public or 3rd-party as software evaluators. This means that the idea can be publicized, because significance is weighted far more heavily to the implementation of the idea.

I hope you weigh this with all the other feedback from actual software engineers (as I am one) with more weight than software attorneys, because we as engineers have an insight to the true nature of software that really takes experience in the field to acquire.

-Jeff