From: Rich Hilliard [e-mail redacted]
Sent: Monday, September 27, 2010 5:57 PM
To: Bilski_Guidance
Cc: [e-mail redacted]
Subject: Comments on Bilski Guidance

I am pleased to submit these comments in response to USPTO request.

Rich Hilliard

Comments on USPTO Interim Guidance in view of *Bilski v. Kappos*

Rich Hilliard r.hilliard@computer.org	IEEE Computer Society, ISO/IEC JTC1/SC7 WG 42: Architecture

The USPTO asks for guidance from the public in response to three questions:

> 1. What are examples of claims that do not meet the machine-or-transformation test but nevertheless remain patent-eligible because they do not recite an abstract idea?
>
> 2. What are examples of claims that meet the machine-or-transformation test but nevertheless are not patent-eligible because they recite an abstract idea?
>
> 3. The decision in Bilski suggested that it might be possible to "defin[e] a narrower category or class of patent applications that claim to instruct how business should be conducted," such that the category itself would be unpatentable as "an attempt to patent abstract ideas." Bilski slip op. at 12. Do any such "categories" exist? If so, how does the category itself represent an "attempt to patent abstract ideas?"

I would like to offer some guidance on these questions, based on 32 years of experience in the areas of computer science, software engineering and software architecting, while working in both the private and public sectors and in several standards-making bodies in computing and information processing (IEEE, ANSI, ISO).

1) Are there claims that do not meet the machine-or-transformation test but remain patent-eligible as they do not recite an abstract idea?

Considering software, there are no such claims which remain patent-eligible because *software is mathematics* at its foundation. This is not only recognized throughout the community of computing professionals, computer scientists and academics, it is a foundation that makes possible advances and innovations in those fields.

This fact has been recognized in the US by numerous courts including the Supreme Court, in relation to the patentability of mathematics and *algorithms*—one form of mathematical expression used in software. *To discover an algorithm, is to make a mathematical discovery.* In this sense, software as mathematics represents a category of abstract ideas, and should not be patentable in itself. The appropriate form of protection of ideas expressed in software is through existing copyright laws: software expression can be protected in the multitude of programming languages in use. Through these programming languages, abstract

ideas are expressed, and only those forms should be protected - via existing copyright provisions.

2) Are there claims that meet the machine-or-transformation test but are not patent-eligible as they recite an abstract ideas?

In regard to this question, I would like to offer a caution with respect to use of the term "machine" which patent examiners should be aware of.

Within the branches of mathematics upon which most software is based, there are a number of abstractions called "machines": Turing machines, automata, virtual machines, ... the term "machine" is unfortunate because it connotes a abstract idea that some patent examiners may confuse with physical machines.

Turing machines, invented by the mathematician Alan Turing in 1937, are an abstract idea embodying how to carry out *any computation expressed as an algorithm*. Turing machines are a fundamental underlying abstraction throughout computing, and existed prior to any "machine" realizing that abstraction.

Virtual machines occur throughout computing with names like: "operating systems," (such as Windows, OS X, Gnu/Linux), "computing platforms" (such as Android) and "the Web." At their essence is an abstract idea for shared operation, inter-communication and commonality - in effect, a "way of doing business" that is made possible using many different physical machine realizations. In the case of the Web that abstract idea is realized through an unknown number of types of machines (including Mac laptops, PC desktops, GNU/Linux tablets, etc.). The nature of the hardware is irrelevant to the essence of the Web. Programming languages, like Java, often include a virtual machine for running programs. We might say that a "virtual machine" is merely an obvious realization of a Turing machine - a refinement of one abstract idea into another for a particular purpose.

Patent examiners should not be confused by these usages of "machine" since they bear no similarity to machines in the traditional sense of a physical device. While the expression of a new virtual machine can be protected by ordinary copyright provisions, allowing software patents for virtual machines as abstract ideas would seriously impede technological innovation.

3) Is there a narrower category of patent claims in "how business should be conducted"?

Within the world of software, I am not aware of any such category. However, today "how business should be conducted" is frequently governed by national, industrial and international standards. The effect of software patents impedes progress not only in the commercial realm, but increasingly in standards-making bodies such as W3C, IEEE, ANSI, and ISO. It is likely to have a major impact in current national initiatives like the Smart Grid approaches to efficient power

distribution. This is because many recent standards, which should promote progress and commerce are "encumbered" by specific software patents.

Conclusion

Software patents harm individuals and society-at-large by restricting our ability to innovate and interoperate in a world increasingly mediated by computers. 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.

The Supreme Court of the United States' decision in *Bilski v. Kappos* draws the boundaries of patent eligibility much more narrowly than commonly understood at the case's outset. The keypoint 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 because software consists only of mathematics, which is not patentable, and because the combination of such software with a general-purpose computer is obvious.