From: Steve Revilak [e-mail redacted]
Sent: Monday, September 27, 2010 10:27 AM
To: Bilski_Guidance
Cc: [e-mail redacted]
Subject: The patentability of software

To whom it may concern:

My name is Stephen Revilak. I'm pursing a Ph.D. in computer science, and I have worked as a professional software developer for over 10 years. I'm writing to you today to voice my opinion that software should not be patentable subject matter.

I believe Bilski v. Kappos demonstrates that the Supreme court expects the boundaries of patent eligibility to be drawn more narrowly than those boundaries have been drawn in the past. The USPTO should exclude software from patent eligibility because software consists only of mathematics, and the combination of such software with a general-purpose computer is obvious. I also believe that software patents do more to hinder innovation than to promote it. I will address each of these positions in the paragraphs that follow.

To support my claim that software consists only of mathematics, I would like to draw on material from a well-known text in theoretical computer science, "Computability, Complexity, and Languages" by Davis, Sigal, and Weyuker [1]. Page 26 presents a simple language for performing computations over the domain of natural numbers. This language consists of the following four statements:

```
V = V + 1
V = V - 1
V = V
IF V != 0 GOTO L
```

Where V is any variable, and L is any label. (A label is marker that refers to a specific statement in a computer program.)

The first four chapters of [1] describe how to combine these statements into more complex building blocks (called "macros"), ultimately showing that this tiny language is sufficient to perform any computation over natural numbers.

Chapter 5 presents a similar language for working with strings. Chapter 5 concludes with a rather fascinating finding: the language over strings and the language over natural numbers are equivalent. It is possible to mechanically translate either language into the other, which means that both languages have equal computational power.

Turing machines are widely regarded as the fundamental model for a general purpose computer. Chapter 6 goes on to show that the languages developed in the previous chapters are equivalent to turing machines. Thus, in the realm of the theory of computation, any computer program can be decomposed into some combination of four mathematical equations:

```
V = V + 1
V = V - 1
V = V
```

```
    IF V != 0 GOTO L
```

The material in [1] requires a good knowledge of mathematics. Ben Klemens' book "Math You Can't Use" [2] offers a less technical, but very compelling argument that all computer software is merely mathematics.

To justify my claim that software patents do more to hinder innovation than to promote it, I would like to focus on one of the most basic and pedestrian tasks that we ask our computers to do: sorting. Nearly every computer program requires the ability to sort data. An address book program needs the ability to sort entries by name. Financial management software needs the ability to sort checks by check number. Search engines need the ability to sort query results by relevance. Sorting is a fundamental component of software programs.

Wikipedia's "Sorting Algorithm" page [3] lists approximately 40 different sorting algorithms, many of which are published in computer science textbooks. I'd like to take a moment to pose a hypothetical question: what if these sorting algorithms were patented? How might such patents affect the development of new software? Let's assume that each algorithm was patented separately (i.e., there is no broad patent that covered the idea of sorting in general). These 40 patent holders could exert a tremendous influence over new software development. If you're an entrepreneur trying to develop and market a medical records program, then you'll need to (a) license an existing sorting technology, (b) develop your own sorting technology, or (c) omit the use of sorting technology from your product.

None of these options are attractive. (a) potentially requires the payment of a large licensing fee, which is burdensome to an entrepreneur; (b) requires research and development effort in an area that's completely unrelated to the core product offering of managing medical records; and (c) limits the usefulness of the final product (which, in turn limits its potential for success in the market). Clearly, none of these options has the effect of promoting innovation.

Of course, there is always option (d): cross your fingers, and hope that no one sues you.

The important point of this exercise is the following: software is built upon a large number of underlying technologies, and sorting is only one example of how this is done. Computer programs can contain hundreds of such "sub-functions", and requiring software developers to research and license each sub-function places a significant burden on them.

Thank you for your time.

Stephen Revilak
Arlington, Massachusetts

---

[1] Computability, Complexity, and Languages: Fundamentals of Theoretical Computer Science, 2nd ed. Martin Davis, Ron Sigal, Elaine Weyuker. ISBN 0122063821.

[2] Math You Can't Use: Patents, Copyrights, and Software. Ben Klemens. ISBN 0815749422.

[3] http://en.wikipedia.org/wiki/Sorting_algorithm, viewed 9/27/2010.