From: Eric Schug [e-mail redacted]
Sent: Saturday, September 25, 2010 5:31 PM
To: Bilski_Guidance
Cc: [e-mail redacted]
Subject: In response to Request for Comments

After review of the Supreme Court Decision in Bilski v. Kappos, firstly wish to congratulate the USPTO for taking an open minded view of the issues and requesting from the general public comments, and not just those who specialize in patent laws, regarding its policy making. The policies of the USPTO effect all US citizens, inventors, businesses, workers, owners, purchasers and suppliers.

The in the modern society we are greatly effected by knowledge, ideas and collaboration of others. Fewer and fewer ideas are transformed in isolated centers like those during Tomas Edison's time or even a generation ago. The most prolific of field in which exchange takes place is within the Information Science field. Two ideas the internet and open source software are the primary causes of its exponential growth.
Review of popular software collaboration sites show that there are hundreds of thousands of people, working on hundreds of thousands of software projects, written in over 50 programming languages. Because all of these projects are publicly available, for modification and review, ideas from each of these projects and be taken and used in other
software. It is not in our best interest, the citizens of the United States to continue to treat software as patentable, novel forms of work.
The shear volume in the number of original software works both in collaboration and in isolation makes the testing for uniqueness and novelty insurmountable, putting a severe burden on the USPTO.

Software is fundamentally the documentation of abstract ideas related to math, science, logic. Ideas which should not be patentable. This is why it is so easily exchanged and collaborated. The ideas and algorithms from one projected can simply be cut and pasted into another project. This no different then editing a text processing document. In fact very similar tools are used to edit software code as those to editing word documents. Even in compiled form, the code

contains just the abstract ideas present in the original form. Translating the code to that form allows the computer to efficiently manipulate and decipher the structure. This is the one and the same as that which is done to a text processing document as it is converted from the human readable version on the screen to the binary encoded version stored on the disk.
Further, for quite some time there have been algorithms which allow for the direct translation of programming languages to natural languages and diagrams used in automatic software documentation. Increasingly the development of natural language processing has allowed for the reverse translation of English text and speech into computer commands or code.
How does one differentiate between document and code? How does one document the implementation without creating the implementation, when the computer can meaningfully read and interpret all human documentation? The first is needed for the free disemination of ideas while the later would be an controlled by the creator under a time limited monopoly.

I have been a user and developer of open source software for quite some time. 90% of the software that I use and create is open source. I am not unique in the regard, many prefer the advantages gained by building on top of others work. Much of the software created and the majority of what runs the internet is open source software (Linux, Apache, etc). The open source is not aided in any way by software patents, and since by definition open source software makes available the document (source
code) describing the process, the society does not benefit from having software patents on open source software. Software patents can only
hider open source. Developers fearful of litigation will write less code, or make it available less often. Most open source code is developed by individuals who do not have the resources to cross check their algorithms against the large numbers of patents. Users fearful of litigation will use less open source software because they do not have the technical knowledge to interpret algorithms and software patents.
Fewer users of open source software will reduce the feedback provided to developers reducing possibilities for collaboration and innovation.

Businesses fearful of litigation provide less funding to open source projects. Businesses spend more on litigation control and on developing in-house solutions rather than on cooperation and advancement. Open source software has greatly enhanced our society and does far more than software patents to create standards, promote openness and exchange ideas.

Developers of software already have a means to protect their investment through the use of Copyright laws. Software patents are not needed for this. Using the method of patenting for IP protection slows the time for development vs using Copyright protection. With Copyright based IP protection software can be conceived created and distributed within one day. Copyrights are immediate when distributed. Patents must first be researched for both related patents and prior art in trade magazines and other documents. Then the patent application must then be created and submitted. Often the patent must be resubmitted do to conflicts with existing patents. This process greatly slows the rate at which new works can be created, and puts the US at a disadvantage compared to other countries without software patents.

Consistant with the ruling in Bilski v. Kappos the USPTO can, and should, exclude software from patent eligibility on the grounds that software consists only of mathematics, which is not patentable, and the combination of such software with a general-purpose computer is obvious and does more to damage collaboration and advancement then to aid.

Eric C. Schug

email: schugschug@gmail.com
Owner and Senior Engineer
Trilogy Scientific, LLC