Theory and reality for software patents: good in concept, not so good in practice

Are software patents bad for developers and researchers? The debate has been raging on this topic as more and more software patents are issued and intellectual property staked out, but the answer ultimately is not as simple as a yes or no. Patents arose as a legal mechanism designed to protect ‘intellectual capital’; that is, to protect the ideas of inventors from being used by others without permission and/or compensation. It was recognized early on that if one wished to provide incentive to innovators, that not all inventors had the means to implement, develop or even produce their invention at the time it was conceived. Without some kind of protection, inventors were reluctant to aggressively pursue their ideas, much less share them with others that might prefer stealing the idea to sharing the glory. And, as a general rule, people often scoff at new ideas until someone proves they are useful, which usually requires someone to invest significant resources such as time, money and effort. In theory, patents provide a valuable protection system for inventors pursuing development of their invention as well as a reward system for commercially valuable ideas.

Modern-day patents, however, are not always of such a noble nature and software patents, a relatively new concept, are becoming more of an issue in bioinformatics (Lesavich, 2000). These patents may eventually wind up as landmines in the path of progress, as there is now clearly a divergence between theory and reality.

The term ‘software patents’ is a little bit of a misnomer since in general, it is not the software that is patented, but the underlying method or algorithmic means of accomplishing an end result. It would not make much sense, for example, to patent your Java code only to have someone circumvent it by writing a slightly different program to do the same thing in C. Software patents are also known as ‘computer implemented inventions’. And there are two major areas whereby software patents pose a potential problem to bioinformatics research: first, part of the problem lies with the criteria by which patents are granted. To obtain a patent for a claimed invention, it is supposed to be new, useful and non-obvious. If just one of these criteria is missing, then—in theory—the patent application should be rejected. Software differs from most other inventions in that end-users never see what was done algorithmically to achieve the end program—programmers are constantly reinventing the wheel and rarely documenting what they did in a publicly searchable format that could be used to establish prior art. And unfortunately each of the criteria for patenting can be subjective, which is reflected in the patent database. For example, it has been pointed out (see http://www.freepatentsonline.com/crazy.html) that at least five patents have been granted for the method of using a laser pointer to amuse a pet (Amiss and Abbott, 1993; Lorenz, 2001; Nathanson and Kersaint, 2001; Allen, 2002; Chelen, 2002). Four of these patents were granted despite the existence of previous patents of the same method and that of a publication predating even the earliest of these patents by over a decade describing a virtually identical method (Stidger and Stidger, 1982). Yet, until the patents are officially overturned, each of these inventors can legally sue infringers, including each other. Lawsuits, of course, cost time and money. While in some cases such as this one, they might be settled relatively quickly, in other cases they may drag on for a long time... which leads us to the second major problem with software patents.

The second problem with software patents, and perhaps the patent system in general, is the issuance of broad and/or vague patents. Of course, the broader the claims, the more likely they are to either be considered obvious or to impinge upon prior art and be rejected at the application stage. Yet, ironically, despite the higher probability of being overturned, vague patents can be even more valuable than specific ones if they are ultimately issued, as they permit the inventor to subsequently lay claim to a broad swath of intellectual territory, including specific ideas they would never have conceived of independently. These patents can become serious stumbling blocks to progress, and the implications of this range from the mundane to the severe. Few people would care if a clever graphical interface never made it to market because of litigation, but what about an algorithm that helps to better understand, diagnose or treat a disease? Fortunately, at least, congress is currently considering a patent reform package that includes a provision to eliminate the injunction associated with patent infringement if the patent holders are not themselves making the product.

These two problems, combined with an explosive growth in the number of software patents, have led to a third problem—defensive patenting. For most companies that actually produce a software product, the motivation is not so much to sue other people for infringing their patent, but to avoid being sued. Even if they can clearly show that they invented something prior to another patent claiming the same invention, litigation can still be costly. The original intent behind their patenting, however, may change—these patents can equally be used offensively. It has been convincingly argued that the biggest beneficiaries of software patents are lawyers.

The tie-in to bioinformatics research should be apparent for software developers: How many patent ‘landmines’ might you be in danger of stepping on when you write your next program? And let us face it, if someone can patent a method for swinging on a swing or the use of a wheel for transportation (Olson, 2000; Hecht, 2002), it may just be a matter of time before someone tries to patent the idea of a For...Next loop (e.g. ‘We describe a method for the iterative algorithmic processing of command statements, the number of iterations being fully customizable and controllable by a user.’)...). Let us just hope the patent office is vigilant if they do. Yet, at the same time, it should be noted that there have been no high-profile cases of bioinformatics research being hindered by patent issues so far. In an odd way, the same shady motivations of software patenting and litigation that put companies at risk protect the academic enterprise. There is normally very little money in suing researchers, after all. So, although we may technically already be in violation of hundreds of patents from the software we have written and published, I am willing to bet most of us will never know about it unless we become filthy rich as a direct consequence. However, the potential threat software
Editorial

patents could pose to bioinformatics research should not be ignored, especially when it comes to the growing array of web services and downloadable analysis programs developed by researchers, which could cut into the sales of a company trying to provide a similar service.

With all these potential problems, why not just end the practice of issuing software patents? Salzberg and Quackenbush argue that we should (Salzberg and Quackenbush, 2006). I would argue that software patents are not inherently evil, and that the potential for profit provides more motivation to pursue the realization of a powerful idea than academic considerations alone. Thus, software patents should not be abolished, but the system reformed. Consider an example: In 1998, Sergey Brin and Larry Page published a paper on a search engine they were working on called Google (Brin and Page, 1998). They filed a patent the same year (Page, 1998), which was granted in 2001. At the time, the reigning search engine giants like Yahoo, Altavista and Webcrawler were already well established, flush with resources—at least compared to an academic start-up that began in a garage. Google is today the world’s most popular search engine, earning that distinction by giving people a superior product. But what would have happened to Google if their idea could have been stolen without legal consequence? First of all, I doubt that investors would have risked their money on an idea that could not be protected but, assuming they did, then I suspect the reigning search engines would have ignored Google when it began. Why change their apparently successful approaches, after all? But around 2000 or so, and after a lot of resources by the Google team had already been committed and their success was growing rapidly, I think the same search engines would have begun re-evaluating the situation and considered adopting Google’s methods in face of this growing threat. They could have used their existing resources to re-implement Brin and Page’s algorithm (thereby avoiding copyright issues) and quash this newcomer before it could grow too large and powerful. And we, the consumers, would likely be worse off, since the value Google offers as a company encompasses many other services besides their search engine (e.g. Gmail, free APIs). Thus, if software patents can be used to protect inventors and investors actively pursing development of a clever idea, then they can be a good thing. So the question remains of how to keep the potential for good and lessen the potential for evil.

First, the criteria for awarding patents should include the intent to pursue development of an invention. Some might argue that it is ok to conceive of an idea, patent it and wait to sue for infringement anyone who actually develops a working system that was only an abstract concept in their patent application. Euphemistically, this would be termed a ‘business strategy’ or ‘strategic intellectual property placement’. I would argue that, while legal, this is completely contrary to the original intents behind patent law. This ‘patent trolling’ is a travesty that has the potential to completely hinder progress for the 20 year lifetime of the patent. And, until recently, another abuse of the (US) patent system enabled progress to be stymied for even longer—a tactic known as ‘submarine patents’ (see http://en.wikipedia.org/wiki/Submarine PATENTS) permitted patent filers to continually revise their application and protect their IP while postponing the actual start date. This was recently abolished by changing patent life from 17 years from issuance to 20 years from initial filing. But this is still a long time to ‘sit’ on an idea. Patent life should be shortened to 5 year terms, with 4 consecutive renewals possible. Each renewal would be contingent upon showing tangible or continued progress in pursuing the idea. People who are actually developing can easily show it, but companies that want to sit on a large patent portfolio will get overwhelmed quickly if they have to constantly pretend they are working. Their former ‘assets’ will become liabilities. Currently, there are patent maintenance fees that must be paid at 3.5 years ($450), 7.5 years ($1150) and 11.5 years ($1900), and this helps slightly. But given that patent-holders pay between $3000 and $60,000, depending upon patent complexity, and average somewhere around $15,000, this is not much disincentive.

Second, despite the apparent fondness for lawyers to speak in a language the rest of us have difficulty understanding, patents should be written clearly and in a language all of us can understand. For example, I can imagine searching the patent database for the presence of the word ‘wheel’, but would never think to search for its legally equivalent phrase ‘circular transportation facilitation device’. Use Occam’s razor liberally—if a simpler phrase can be used in place of a more complex one, force the inventors to use it and reject their patent if they do not. This will help reduce the vague patents and save litigation time.

Third, because it is harder to search for previously implemented algorithms, the bar for awarding software patents should be raised. It is hard enough to compile prior art in non-software fields, but there is no searchable code archive and even if there were, there are many different programmatic ways of accomplishing the same end result in many different languages. Patentable software inventions should be substantially different from prior art and very non-obvious. If you task 10 programmers to solve the same general problem the invention claims to solve, and at least half come up with the same solution in a day, it is obvious.

Fourth, perhaps the patenting system could borrow something from the academic community and begin peer-reviewing patents, requesting several independent evaluations from outside experts. One of the problems with the patent office is that their expertise and time is limited compared with the number of applications they receive. Peer-reviewers would be paid for their time, make their names known on the application and their roles be limited to providing advice to the examiner rather than making the final decision (to avoid conflicts of interest). After turning their comments in, they would also be able to see and comment on all the other reviewers. This way, if there are any particularly good or bad reviews, then this helps highlight them by further peer-reviewing the peer-reviewers. Finally, perhaps journals should enforce a difficult choice for most academics—to publish OR patent, but not both. A patent is a publication, after all, even if not peer-reviewed.

We live in the digital age, and most of the new innovations coming out are computer related. The call to end software patents is growing, but doing so may be a Faustian bargain. How will we protect and incentivize the next great idea to become a reality—to move from paper to practice? Clearly, though, the list of problems associated with software patents is growing and the list of solutions needs to catch up.

ACKNOWLEDGEMENTS

I would like to thank Steven Salzberg and John Quackenbush for a lively and valuable discussion on the merits, ethics and issues
of software patents and Anil V. Gollahalli for his helpful legal review.

Conflict of Interest: The author is listed as an inventor or co-inventor on three software-related patents, all of which are pending approval by the USPTO.

Jonathan D. Wren
Advanced Center for Genome Technology
Department of Botany and Microbiology
The University of Oklahoma
101 David L. Boren Boulevard
Rm 2025, Norman, OK 73019, USA

REFERENCES


Salzberg,S.L. and Quackenbush,J. (2006) It is time to end the patenting of software. Bioinformatics, in press.