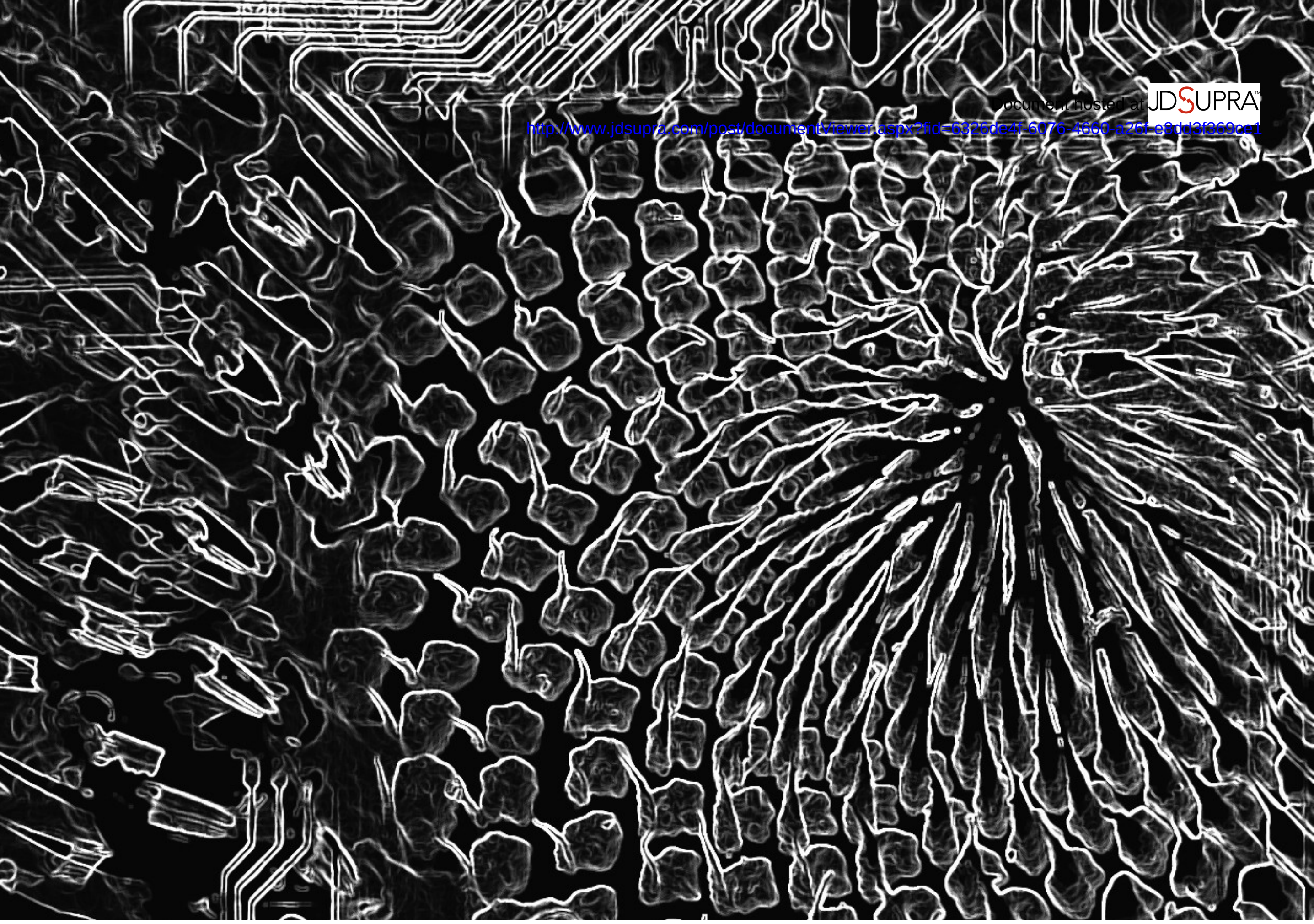


THE MASTERS WOULD BE *jealous*

Whether you're are a brush stroke away from brilliance or still sketching, we're here to help.

Give us a call at 1.800.426.5801 if you're having a breakthrough, or visit us at www.lanepowell.com.

LANE POWELL
ATTORNEYS & COUNSELORS

KNOW THE LANDSCAPE™

Culture clash

Proprietary and open software have developed in parallel in the US over the past two decades. **Craig Bachman** and **Anne Glazer** of **Lane Powell** examine some legal intersections between the two models

From the early 1980s a cultural divide developed in the software community between proponents of proprietary software and advocates of free or open source software. The reasons for this separation of views are many. They include political, philosophical, economic, technical and many other factors. Over time the various camps have learned to co-exist on a number of levels. The actual number of legal disputes centred on these very different approaches to software development and use has been far less than one might have expected. Still the functional aspects of these two distinct approaches to software development have and will continue to result in differences likely to be addressed through developments in intellectual property law. To discuss some of these developments past and future we need a common context.

What is proprietary software?

Proprietary software is software with restrictions on using, copying and modifying as enforced by the owner. Restrictions on use, modification and copying are achieved by both legal and technical means. Technical means include releasing machine readable binaries to users and withholding the human readable source code. Legal means can involve trade secrets, non-disclosure agreements, software licensing, copyright and patent law.

What is open source software?

The most widely accepted approach to describing open source software is that of the Open Source Initiative. Open source means the user of the software must have, or have access to, the source code, but it also means more. The user also has the rights to redistribute, aggregate, modify and create derivative works based on the software. Open source software may be subject to some significant restrictions on use. Derived works may be required to carry a different identity, there may be no restrictions against use in particular fields of endeavour or particular product uses. Importantly under the most widely accepted definition the licence, derived works must not place restrictions on other software that is distributed along with the licensed software. For example, the licence must not insist that all other programs distributed on the same medium must be open-source software (http://www.opensource.org/docs/osd).

What is free software?

The term free software has developed largely through the work of Richard Stallman and the Free Software Foundation (FSF). Stallman's approach to the definition and use of non-proprietary software is broader both technically and philosophically:

> Free software is a matter of liberty, not price. To understand the concept, you should think of free as in free speech, not as in free beer. Free software is a matter of the users' freedom to run, copy, distribute, study, change and improve the software. More precisely, it refers to four kinds of freedom, for the users of the software:
>
> - The freedom to run the program, for any purpose (freedom 0).
> - The freedom to study how the program works, and adapt it to your needs (freedom 1). Access to the source code is a precondition for this.
> - The freedom to redistribute copies so you can help your neighbour (freedom 2).
> - The freedom to improve the program, and release your improvements to the public, so that the whole community benefits (freedom 3). Access to the source code is a precondition for this. http://www.gnu.org/ philosophy/free-sw.html

Stallman's approach and views are important because they have led to the development of one of the most widely used open software licences.

History of open source software

To appreciate the cultural relationships of the open source and proprietary software communities some understand-

ing of their history is important. There are two primary threads in the history of the open source movement. The first centres on Richard Stallman, the Free Software Foundation, and the efforts that led to the GNU General Public Licence (GPL) and the software released under the GPL, particularly the Linux kernel. The second thread centres on the University of California at Berkeley and its role in developing the Unix operating system and led to the BSD Unix family of programs and licences. Both threads trace their roots back into the 1970s and 1980s. A more recent development occurred in 1998 when Netscape released its browser software under an open source licence through an organization known as Mozilla.org.

The BSD licence, while similar in a number of important ways to the GPL, did not require that the derivative works be subject to the same terms as the initial BSD licence. The BSD has served as a model for a number of other open source licences.

In order to release the Netscape browser under an open source licence such as the GPL, each underlying licensor of software used in the browser would have to agree to release their incorporated code under the open source licence chosen by Netscape. The prospect of getting those agreements was unlikely. As a result Netscape decided that the best approach would be to write its own open source licence. That new licence would attempt to address Netscape's specific issues while keeping within the general requirements of the open source types of licences and making some "improvements" to the language and coverage of the licence.

Netscape's initial efforts illustrate the tendency of commercial developers moving into open source to want to rewrite standard open source licences. Second, Netscape's original effort to broadly rewrite the licence showed that open source is truly a community-based project and the community will react to and pressure anyone who wants to change open source licences. Open source programmers can also withhold efforts on projects if the licence terms for those projects violate the spirit of the open source movement. Netscape wanted a new licence to cover both the existing licensed software in its program and to allow it to retain certain rights to release proprietary commercial versions of software developed as part of the proposed Netscape open source project. Netscape's cause was helped because it enlisted the assistance of the open source community and, more important, it listened to the comments of the community and worked with it. After reaching a compromise form of licence acceptable to the open source community, Netscape developed what is now known as the Mozilla Public Licence. This licence now serves as an important model of an open source licence in situations where commercial software or commercial development are involved.

Taxonomy of open source licences

While a standard commercial proprietary software licence is focused on protecting the copyright interests of the owner by restricting the uses of the software, open source licences place minimal restrictions on the use of software in order to "free" the software. The open source licences have a common requirement that source code be made available and that users of the software have the right to make derivative works. The licences all disclaim warranties and many make an effort to limit liabilities. Four broad categories are the GPL or "free" family of licences, the BSD set of licences, including the MIT X licences, the Mozilla Public Licence, and, finally, other non-GPL and "commercial" licences. The categories can be distinguished in a number of ways, but the most important distinction is in the way they address the issue of permitting derivative works to be later made proprietary.

The issue of software patents

Complex structures such as the internet depend on many technological standards. About 30 standards underlie every click of a mouse in a web browser from the HTML file format through the TCP/IP communications protocol, all the way down to a standard specifying the thickness of the gold plating on an ethernet cable. If just five of these standards each contain a single patent bearing a 5% royalty, profit potential for any small company engaging in internet software development would be eliminated.

Today's computer industry standards increasingly include technology that may be covered by a software patent. A patent owner can demand a royalty from all parties that implement the patented principle. Often there is no way to implement a standard without making use of a particular patented principle. This effectively gives the patent holder control of who will implement a standard containing his patented principle. Patents may become embedded in industry standards in a number of ways. They may be knowingly embedded in the standard as it is being created, or they may be submarine patents, unknowingly part of the standard until they surface after the standard is already in wide use. A pernicious patent holder can engage in patent farming: influencing a standards organization to use a particular principle covered by a patent. In the most deceptive form of patent farming, the patent holder encourages the standards organization to make use of a principle without revealing the existence of a patent covering that principle. Later the patent holder demands royalties from all implementers of the standard.

Recent court cases have involved patent farming or submarine patents in standards. The *Eolas* case is a good illustration of a submarine patent (*Eolas Technologies Inc v Microsoft Corp*, 399 F3d 1325, 1339 (CA Fed 2005)). An inventor held a patent on a widely used feature of HTML, the standard that describes the format of

web pages. The patent holder did not make his patent known for years, and then sued Microsoft for use of his principle. The patent holder was awarded a judgment by the court. Over time companies using alternate technology that did not use the principle of the Eolas patent developed non-infringing work-arounds.

The *Rambus* case is a good illustration of patent farming (*Rambus v Infineon*, 318 F3d 1081, (C A Fed 2003)). A manufacturer allegedly influenced an organization attempting to standardize computer memory to use its technology. The manufacturer did not disclose that it had a patent application in process on the same technology. Years later, the manufacturer started bringing patent infringement lawsuits against all implementers of the memory standard. Reactions of industry leaders resulted in movement away from the Rambus technology, enhancement of alternate technologies and collaboration among industry leaders ultimately leading to a large antitrust investigation and enormous fines and guilty pleas.

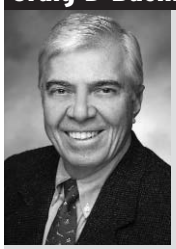
The impact of such a situation is most severe for open source developers. Those developers generally do not charge a royalty for their software, and thus generally cannot afford to pass on royalties to patent holders. Open source lives on collaboration, and its collaboration forms only when all parties have the same rights. The presence of a required royalty payment for implementation of a patented principle would probably abort any open source development in that area. Since today's software patents are written to be deliberately vague in order to have the widest possible scope of enforcement, it is a truism that any significant work of software infringes upon some patent. The patent situation is bad enough when applied to the general field of software. When applied to standards, it specifically restricts the interoperation of programs and communication between them, because those are the subjects of computer industry standards. Thus, software patents can be used to prohibit interoperability.

Obviousness and prior art

The cultural gap between the open source and proprietary communities can be a factor creating forces which in the end may reduce the grip of proprietary holders on legal ground they have staked out for many years. In the recently decided patent case *KSR International v Teleflex* the Supreme Court reminded us that patented matter might be obvious to a person of ordinary skill in the art even though the often used teaching, suggestion, obviousness shortcut test for obviousness is not met (550 US ___, 127 S Ct 1727 (2007)). An open source advocacy group, the Electronic Frontier Foundation, was among those filing *amicus* briefs supporting the logic of the Supreme Court's decision. The EFF noted that the experience of the open source community was a particularly relevant consideration because it provides an example of a climate where development takes place in the open and it may well be obvious to many to try the patented principle.

Similarly, the open source experience suggests that in open development contexts relevant and potentially invalidating prior art may exist outside the patent office file.

Craig D Bachman



Craig D Bachman, co-chair of the Lane Powell intellectual property and technology practice group, practises primarily in the field of intellectual property and antitrust litigation. He has been a trial lawyer since 1978, representing clients in trials involving patent, trade mark, copyright, antitrust, trade regulation, insurance practices and professional malpractice. He also counsels clients concerning strategic management of patent, trade mark and other IP assets, and in distribution, branding and marketing. Bachman's technology background was in biochemistry, and he has long had interest in the economic issues surrounding technology development. He earned his JD from Lewis and Clark College Northwestern School of Law and his BS from Portland State University. He has been named in *Best Lawyers in America* and *Oregon Super Lawyers* for intellectual property, and he frequently writes and speaks about intellectual property, antitrust and trial techniques.

Anne W Glazer

Anne W Glazer, co-chair of the Lane Powell intellectual property and technology practice group, practises primarily in the fields of trade mark, copyright, licensing and contracts. As head of Lane Powell's trade mark team, she is especially skilled at counselling and representing companies in all aspects of trade mark law and managing trade mark portfolios. She also counsels and represents companies in the areas of e-commerce, the internet, distribution, marketing and advertising. Glazer has significant experience in IP and unfair competition litigation, including computer software, trade secrets, trade marks and copyrights. She earned her JD with honours and her BA *magna cum laude*, both from the University of Washington where she was a founding member of the *Pacific Rim Law & Policy Journal*. Ms. Glazer has been named in *Best Lawyers in America* for the past seven years. She is a frequent speaker and author on intellectual property and internet topics.