19

Public Licenses: Open Source, Creative Commons and IP Pledges

SUMMARY CONTENTS

This chapter discusses licenses that are granted to the public at large, typically without monetary compensation. Like consumer EULAs, these "public" licenses are made available to potential users online, do not require signature, delivery or formal execution, and are generally effective automatically upon the user's download or use of the licensed content. Despite their relative informality, such licenses underlie vast quantities of online content, computer software and even patent rights today. Below, we discuss the history, motivations and strategies of three distinct types of public licensing: Creative Commons online content licenses, open source software (OSS) licenses and patent pledges.

19.1 CREATIVE COMMONS AND OPEN CONTENT LICENSING

THE CREATIVE COMMONS
LAWRENCE LESSIG, 64 *Montana Law Review* 1, 10–13 (2004)

In the beginning of the Internet, the architecture of the Internet disabled any ability to control the distribution of copyrighted works. That meant that if we had this triad among all, some, and none, the effective protection of the original Internet was none. The architecture meant that copyright was not respected because anybody could copy and perfectly distribute any copyrighted work without control.

That extreme begot another: the terror of the copyright industry, which in 1995, in response to the Internet launched a campaign to change the technical and legal infrastructure that defined the Internet, to change the Internet from an architecture of no control into an architecture of total control. So again, instead of a triad, we have increasingly an architecture of total control over everything. We have thus moved from one extreme to the other: from the extreme of total freedom to total control, and this is the shift the law is encouraging.

https://doi.org/10.1017/9781009049436.020 Published online by Cambridge University Press

We're setting up a regime that thinks as if the world is either one or the other when it is in fact neither. Some want total control, some want no control, but most want this balance in the middle. Not "all rights reserved," not "no rights reserved," but increasingly the idea of "some rights reserved." My content is out there, I want you to respect it in some ways, but I want you to use it in lots of ways that traditional "all rights reserved" models would not permit.

Enter an organization that I have helped start called the Creative Commons. Think of it as pushing, as another founder, James Boyle, describes it, as a kind of environmentalism for culture. The idea here is that we need to build a layer of reasonable copyright law, by showing the world a layer of reasonable copyright law resting on top of the extremes. Take this world that is increasingly a world by default regulating all and change it into a world where once again we can see the mix between all, none, and some, using the technology of the Creative Commons.

This change is done through the voluntary action of individuals – creators, content owners. They take voluntary action by marking their content with a tag that expresses a kind of freedom. They use these tags then to build a kind of balance into the system to restore this reasonableness into the system by giving people a way to say, "I don't believe in this extremism."

For example, if you go to the Creative Commons Web site (http://creativecommons.org), you are given a very simple choice by which you can select the freedoms you want to grant. You can say, "I want people to give me attribution or not," or you can say, "I want people to use this for commercial use or not," or you can say, "I want to allow people to modify this or not," or you can use what we call a "share alike" license that says, whatever freedoms you got from me, you have to pass on to someone else.

Once you make these selections, the technology then produces a license that is comprised of three separate layers. One layer is a human-readable version of that license. Another is the lawyer-readable version of the license – the license. And a third layer is a machine-readable version of the license, which enables computers to understand what freedoms you are granting.

These three layers live together in a "Creative Commons" tag. And in four months, more than 400,000 pages have appeared on the Internet linking back to these licenses. Four hundred thousand have said, we believe in a kind of freedom associated with our content that is not the extreme.

Now we want this 400,000 to turn into 10,000,000. Because if there are 10,000,000 people out there who say we don't believe in the extremes, then this debate is no longer a debate between copyright owners and anarchists. Instead, it is increasingly a debate between extremists and those who believe in a tradition that expresses a freedom more fundamental. Beyond the permissions of fair use, these licenses give people ways to say go ahead, sample me, share me, copy me, liberate me, and together they restore something of balance in this debate. And this balance, we believe, will enable a different kind of creativity: creativity built upon a tradition of building upon the works of others, freely. A free culture, not the permission culture that our law has produced.

There is an extraordinary potential enabled by a technology that is increasingly threatened and destroyed. The potential for a different, critical, democratic creativity, is increasingly being forced into last century's model for doing business.

It is a world where the dinosaurs have been given the power to control evolution. A system where last century's powerful has the ability to veto next century's innovators. In such a world, creativity and innovation die. The free culture that defined our tradition has been eroded, not by idealists, but by lobbyists. The free culture that we have lived under now is under threat. And we have an obligation, all of us, to engage in this practice to enable this freedom again.

Lessig's vision of ten million Creative Commons (CC) tags has been more than fulfilled. The CC website today claims that more than 500 million online images are available under CC licenses. As Lessig and others intended, the appeal of the CC licensing system is its simplicity and its intuitiveness. Users can choose to apply one of six different combinations of four different licensing options to their works. Each option is described in simple, plain language and identified by an intuitive icon.

Thus, if I wish to post a photo to a social media site and make it available for anyone else to use for any purpose so long as they give me credit (attribution), I can tag the photo with the "CC BY" symbol, and the CC Attribution license will apply. If I also wish to stipulate that my photo cannot be modified in any way, then I can tag it with the "CC BY ND" (Attribution, No Derivatives) license.[1] If I want to be sure that my photo remains free for all to use, even if someone incorporates it into a proprietary database or website, then I can add the "SA" (Share Alike) tag. And if I wish to prohibit commercial uses (e.g., using my photo in a corporate ad), then I can use "NC" (Non-Commercial). As shown in Figure 19.1, there are only six permitted combinations of these four licensing tags (out of fifteen possible combinations), reflecting the designers' views of the most frequent and logical types of uses that should be permitted.

The CC suite of licenses appears simple, but a sophisticated legal structure underlies its streamlined user-facing tags. That is, the tag "CC BY ND" does not itself convey a license to the user. Rather, when a tag is attached to an online image or other content, it includes a hyperlink to a more comprehensive licensing agreement that is hosted on CC's website. For example, the full text of the CC BY NC ND 4.0 license can be found at https://creativecommons.org/licenses/by-nc-nd/4.0/legalcode.

By many measures, the CC licensing framework has been phenomenally successful. Professor Jane Ginsburg points to four important design features that have contributed to the success of the CC model: its overall simplicity, its extension of credit to authors (included in each of the six permitted licenses), its ability to authorize use of the licensed content instantly and forever, and its potential to expand distribution of a work through search engines.[2] These features have made CC licensing a standard feature of online platforms and social media sites.

Notes and Comments

1. *Which rights reserved?* In Lessig's view, why is a third option necessary for copyrighted material in addition to "all rights reserved" and "no rights reserved"?
2. *Public licenses*. The CC licenses are "public" licenses. That is, they are not specifically negotiated between copyright owners and users, but are publicly posted and can be "accepted" by

[1] The CC licenses also include a version number reflecting updates that have been made over the years. The current version is 4.0, so a full CC tag would read: "This work is licensed under a *CC BY 4.0 license*."

[2] Jane C. Ginsburg, Authors' Transfer and License Contracts under US Copyright Law in Research Handbook on Intellectual Property Licensing 3, 23 (Jacques de Werra, ed., Edward Elgar, 2013).

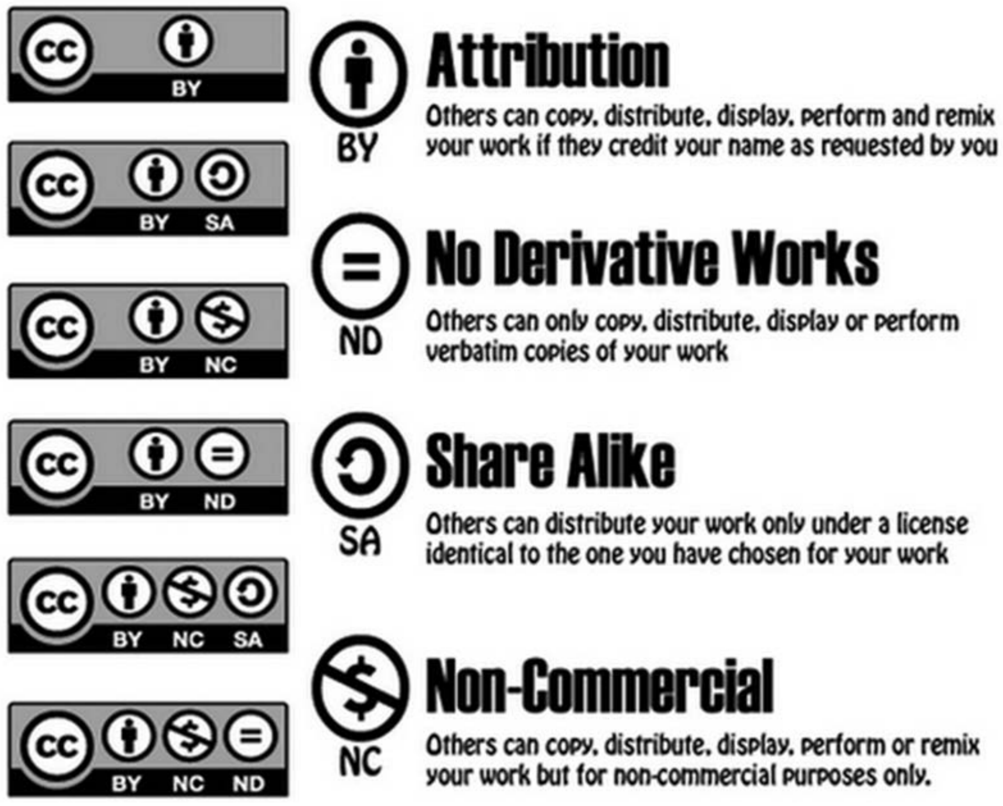

BY
BY SA
BY NC
BY ND
BY NC SA
BY NC ND
Attribution
Others can copy, distribute, display, perform and remix your work if they credit your name as requested by you
BY
No Derivative Works
Others can only copy, distribute, display or perform verbatim copies of your work
ND
Share Alike
Others can distribute your work only under a license identical to the one you have chosen for your work
SA
Non-Commercial
Others can copy, distribute, display, perform or remix your work but for non-commercial purposes only.
NC

FIGURE 19.1 The Creative Commons suite of licenses.

anyone who wishes to use the licensed content. Thus, the introduction to the CC BY NC ND 4.0 license reads as follows:

By exercising the Licensed Rights (defined below), You accept and agree to be bound by the terms and conditions of this Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International Public License ("Public License"). To the extent this Public License may be interpreted as a contract, You are granted the Licensed Rights in consideration of Your acceptance of these terms and conditions, and the Licensor grants You such rights in consideration of benefits the Licensor receives from making the Licensed Material available under these terms and conditions.

How consistent is the CC approach to that recognized by courts in the context of consumer EULAs and terms of use discussed in Chapter 17? Why aren't more IP licenses structured as public licenses?

3. *Permitted combinations*. Why did the designers of the CC licenses only permit six out of fifteen possible combinations of the four licensing tags? Can you think of any useful combinations beyond the six permitted ones?
4. *The importance of attribution*. In one early survey of CC licenses, 98 percent of users selected the BY attribution requirement – far more than any of the other licensing options. For this reason, when it revamped its licensing options, CC included the "BY" requirement in all options (i.e., there is no CC licensing option that omits "BY"). Why do you think attribution is so important to content creators?

CC is fairly flexible when it comes to specifying how attribution must be made. The license states:

You may satisfy the conditions in Section 3(a)(1) in any reasonable manner based on the medium, means, and context in which You Share the Licensed Material. For example, it may be reasonable to satisfy the conditions by providing a URI or hyperlink to a resource that includes the required information.

Is this flexible approach ideal? How might it be abused?

5. *Sharealike and Copyleft*. The Share Alike or "SA" feature of CC licenses resembles the controversial "copyleft" approach to some OSS software promoted by the Free Software Foundation and others (see Section 19.2). As such, CC SA is probably the least commonly used variant of the CC licensing suite. What is the rationale for imposing an SA licensing requirement? Why do you think it is not widely used? Keep these issues in mind as you review Section 19.2 and 19.3.
6. *Applications*. The core application for CC licenses is digital content, particularly images such as photographs, drawings and artworks. But the CC licenses are general copyright licenses that are not strictly limited to visual images. It is not difficult to imagine how CC licensing could be used for text – blog posts, academic articles, short stories, poems. But what about music? In the mid-2000s, CC created three music sampling licenses,[3] but those have since been discontinued in favor of the general suite of six licenses. Likewise, CC has experimented with licenses for scientific data, though that project has also been discontinued. And despite the growth of OSS licensing (see Section 19.2), CC licenses are seldom used for software. Why does CC licensing appear to be limited to visual images? Are there other logical expansions for the use of CC licensing?
7. *Database rights*. As discussed in Section 18.1, databases, *per se*, are not protected under US intellectual property law, though they are protected in the EU and other jurisdictions. Thus, the CC licenses, which are intended to apply internationally, include a provision stating that "Where the Licensed Rights include Sui Generis Database Rights that apply to Your use of the Licensed Material," the licenses granted with respect to copyrights are also granted with respect to those database rights. Under what circumstances do you think that this provision could become important?
8. *CC 0*. In addition to its suite of licenses, CC also permits users to select a CC 0 or "No Rights Reserved" tag for their content. As CC explains,

CC 0 enables scientists, educators, artists and other creators and owners of copyright- or database-protected content to waive those interests in their works and thereby place them as completely as possible in the public domain, so that others may freely build upon, enhance and reuse the works for any purposes without restriction under copyright or database law.

Why was the CC 0 option necessary? Does this option conflict with Lessig's original plan to create a system that offered options other than "all rights reserved" and "no rights reserved"?

9. *Choice of law*? The CC licenses do not include an express choice of law provision (see Section 11.3). Why do you think this term was omitted?
10. *The business of CC*. Creative Commons is not a governmental body or an international organization, but a nonprofit corporation based in the United States, with local chapters

[3] See Michael W. Carroll, *Creative Commons and the New Intermediaries*, 2006 Mich. St. L. Rev. 45, 47–48 (2006).

around the world. Is it advisable to entrust the licensing structure for so much online content to a single private organization? What would happen to CC licenses if the Creative Commons corporation were to be liquidated or simply disappear? Is there any viable competitor to CC today? Is there a need for one, now that the CC licenses are published and available online? Why is a legal entity needed at all for a self-executing licensing framework?

19.2 THE OPEN SOURCE PHENOMENON

As discussed in Section 18.2.1, a computer program's "source code" is a version of the program written in a human-readable programming language such as C++, Perl, BASIC or Fortran. Most proprietary software is licensed and distributed in object code or executable form. But beginning in the 1970s, a group of software developers in Cambridge, Massachusetts, began to make their source code publicly available too. This trend began the "free software" or "open source software" (OSS) movement, which today is at the heart of a multi-billion-dollar industry.

19.2.1 *Origins: The Free Software Movement*

The excerpt below is by Richard Stallman, who is generally credited as the father of the free software movement.

THE GNU OPERATING SYSTEM AND THE FREE SOFTWARE MOVEMENT
RICHARD STALLMAN, *Open Sources: Voices from the Open Source Revolution* (1999)

When I started working at the MIT Artificial Intelligence Lab in 1971, I became part of a software-sharing community that had existed for many years. Sharing of software was not limited to our particular community; it is as old as computers, just as sharing of recipes is as old as cooking. But we did it more than most.

We did not call our software "free software," because that term did not yet exist, but that is what it was. Whenever people from another university or a company wanted to port and use a program, we gladly let them. If you saw someone using an unfamiliar and interesting program, you could always ask to see the source code, so that you could read it, change it, or cannibalize parts of it to make a new program.

The situation changed drastically in the early 1980s when Digital discontinued the PDP-10 series. The modern computers of the era, such as the VAX or the 68020, had their own operating systems, but none of them were free software: you had to sign a nondisclosure agreement even to get an executable copy.

This meant that the first step in using a computer was to promise not to help your neighbor. A cooperating community was forbidden. The rule made by the owners of proprietary software was, "If you share with your neighbor, you are a pirate. If you want any changes, beg us to make them."

One assumption is that software companies have an unquestionable natural right to own software and thus have power over all its users … Interestingly, the U.S. Constitution and legal tradition reject this view; copyright is not a natural right, but an artificial government-imposed monopoly that limits the users' natural right to copy.

Another unstated assumption is that the only important thing about software is what jobs it allows you to do—that we computer users should not care what kind of society we are allowed to have.

A third assumption is that we would have no usable software (or would never have a program to do this or that particular job) if we did not offer a company power over the users of the program. This assumption may have seemed plausible, before the free software movement demonstrated that we can make plenty of useful software without putting chains on it.

If we decline to accept these assumptions, and judge these issues based on ordinary common-sense morality while placing the users first, we arrive at very different conclusions. Computer users should be free to modify programs to fit their needs, and free to share software, because helping other people is the basis of society.

So I looked for a way that a programmer could do something for the good. I asked myself, was there a program or programs that I could write, so as to make a community possible once again?

The answer was clear: what was needed first was an operating system. That is the crucial software for starting to use a computer. With an operating system, you can do many things; without one, you cannot run the computer at all. With a free operating system, we could again have a community of cooperating hackers—and invite anyone to join. And anyone would be able to use a computer without starting out by conspiring to deprive his or her friends.

I chose to make the system compatible with Unix so that it would be portable, and so that Unix users could easily switch to it. The name GNU was chosen following a hacker tradition, as a recursive acronym for "GNU's Not Unix."

The term "free software" is sometimes misunderstood—it has nothing to do with price. It is about freedom. Here, therefore, is the definition of free software. A program is free software, for you, a particular user, if:

You have the freedom to run the program, for any purpose.

You have the freedom to modify the program to suit your needs. (To make this freedom effective in practice, you must have access to the source code, since making changes in a program without having the source code is exceedingly difficult.)

You have the freedom to redistribute copies, either gratis or for a fee.

You have the freedom to distribute modified versions of the program, so that the community can benefit from your improvements.

In January 1984 I quit my job at MIT and began writing GNU software. Leaving MIT was necessary so that MIT would not be able to interfere with distributing GNU as free software. If I had remained on the staff, MIT could have claimed to own the work, and could have imposed their own distribution terms, or even turned the work into a proprietary software package.

I began work on GNU Emacs [a text editing program] in September 1984, and in early 1985 it was beginning to be usable … At this point, people began wanting to use GNU Emacs, which raised the question of how to distribute it. So I announced that I would mail a tape to whoever wanted one, for a fee of $150. In this way, I started a free software distribution business, the precursor of the companies that today distribute entire Linux-based GNU systems.

If a program is free software when it leaves the hands of its author, this does not necessarily mean it will be free software for everyone who has a copy of it. For example, public domain software (software that is not copyrighted) is free software; but anyone can make a proprietary modified version of it. Likewise, many free programs are copyrighted but distributed under simple permissive licenses that allow proprietary modified versions.

The goal of GNU was to give users freedom, not just to be popular. So we needed to use distribution terms that would prevent GNU software from being turned into proprietary software. The method we use is called "copyleft."

Copyleft uses copyright law, but flips it over to serve the opposite of its usual purpose: instead of a means of privatizing software, it becomes a means of keeping software free.

The central idea of copyleft is that we give everyone permission to run the program, copy the program, modify the program, and distribute modified versions—but not permission to add restrictions of their own. Thus, the crucial freedoms that define "free software" are guaranteed to everyone who has a copy; they become inalienable rights.

For an effective copyleft, modified versions must also be free. This ensures that work based on ours becomes available to our community if it is published. When programmers who have jobs as programmers volunteer to improve GNU software, it is copyleft that prevents their employers from saying, "You can't share those changes, because we are going to use them to make our proprietary version of the program."

The requirement that changes must be free is essential if we want to ensure freedom for every user of the program. The companies that privatized the X Window System usually made some changes to port it to their systems and hardware. These changes were small compared with the great extent of X, but they were not trivial. If making changes was an excuse to deny the users freedom, it would be easy for anyone to take advantage of the excuse.

A related issue concerns combining a free program with non-free code. Such a combination would inevitably be non-free; whichever freedoms are lacking for the non-free part would be lacking for the whole as well. To permit such combinations would open a hole big enough to sink a ship. Therefore, a crucial requirement for copyleft is to plug this hole: anything added to or combined with a copylefted program must be such that the larger combined version is also free and copylefted.

The specific implementation of copyleft that we use for most GNU software is the GNU General Public License, or GNU GPL for short.

As interest in using Emacs was growing, other people became involved in the GNU project, and we decided that it was time to seek funding once again. So in 1985 we created the Free Software Foundation, a tax-exempt charity for free software development. The FSF also took over the Emacs tape distribution business; later it extended this by adding other free software (both GNU and non-GNU) to the tape, and by selling free manuals as well.

The free software philosophy rejects a specific widespread business practice, but it is not against business. When businesses respect the users' freedom, we wish them success.

Selling copies of Emacs demonstrates one kind of free software business. When the FSF took over that business, I needed another way to make a living. I found it in selling services relating to the free software I had developed. This included teaching, for subjects such as how to program GNU Emacs and how to customize GCC, and software development, mostly porting GCC to new platforms.

FIGURE 19.2 Richard Stallman, founder of the free software movement, speaking in Oslo as Saint IGNUcius in 2009.

Today each of these kinds of free software business is practiced by a number of corporations. Some distribute free software collections on CD-ROM; others sell support at levels ranging from answering user questions to fixing bugs to adding major new features. We are even beginning to see free software companies based on launching new free software products.

Watch out, though—a number of companies that associate themselves with the term "Open Source" actually base their business on non-free software that works with free software. These are not free software companies, they are proprietary software companies whose products tempt users away from freedom. They call these "value added," which reflects the values they would like us to adopt: convenience above freedom.

Teaching new users about freedom became more difficult in 1998, when a part of the community decided to stop using the term "free software" and say "open source software" instead.

Some who favored this term aimed to avoid the confusion of "free" with "gratis"—a valid goal. Others, however, aimed to set aside the spirit of principle that had motivated the free software movement and the GNU project, and to appeal instead to executives and business users, many of whom hold an ideology that places profit above freedom, above community, above principle. Thus, the rhetoric of "Open Source" focuses on the potential to make high quality, powerful software, but shuns the ideas of freedom, community, and principle.

We can't take the future of freedom for granted. Don't take it for granted! If you want to keep your freedom, you must be prepared to defend it.

Notes and Questions

1. *Software and morality*. Stallman's rhetoric is steeped in notions of morality and justice. Is this moralistic attitude surprising when discussing a field such as software development? Do developers of automotive engines, chemical solvents or even chemotherapy agents speak in the same terms about their work? Why is software different? Would the software world today

look different if Richard Stallman had simply moved on to a different project at MIT instead of beginning to develop free software? Would OSS have emerged as a market phenomenon in any event?

2. *Nondisclosure*. Stallman takes great offense at the nondisclosure agreement that he was required to sign. Why?
3. *Assumptions of the software industry*. What three pre-existing assumptions does Stallman posit about the software industry? Do you think that Stallman's depiction of the realities of the software industry of the 1970s were accurate? Was his response a sensible reaction to these realities?
4. A *question of terminology*. Why does Stallman object to the use of the term "open source software"? Why do you think that many preferred this term to "free software"? Consider these questions when you read Section 19.2.2.
5. *Copyleft*. What is "copyleft"? Why does Stallman view it as fundamentally important to free software? Why does Stallman believe that it is necessary that the GPL be applied not only to redistribution of GPL programs, but to *modifications* of those programs? What does he seek to avoid? Consider these issues as you read the next section and the details of the GPL's copyleft provisions.
6. *Free software versus the public domain*. Somewhat surprisingly, Stallman did not argue that software should be contributed to the public domain. He explains that "public domain software (software that is not copyrighted) is free software; but anyone can make a proprietary modified version of it." What did he mean? Why did he prefer copyleft to the public domain?
7. *Value-added*. What does Stallman describe as "value-added" software and why does he object to it? How does this differ from the paid services that Stallman himself provided with respect to software?

19.2.2 *Defining Open Source Software*

The year 1998 was a watershed in the OSS world. The Linux operating system, created in 1991 by a twenty-one-year-old Finnish undergraduate named Linus Torvalds, was quickly becoming the operating system of choice for corporate enterprises. In that year, database vendors Oracle, Sybase and Informix all announced Linux-compatible products, and Torvalds appeared on the cover of *Forbes* magazine.[4] IBM announced that it would distribute and support the OSS Apache web server. Red Hat, a company devoted to OSS software distribution and support, was formed with backing from Intel and Netscape. And Netscape itself announced that it would release the source code for its popular Navigator web browser.

That year also saw the formation of the Open Source Initiative (OSI), an educational, advocacy and stewardship organization dedicated to open software development. The organization grew out of a February 1998 meeting in Palo Alto, California, shortly after the Netscape announcement. Among the organizers of the meeting was Eric Raymond, the author of a 1997 manifesto on open software development titled *The Cathedral and the Bazaar*. As explained on the OSI website:

> The conferees believed the pragmatic, business-case grounds that had motivated Netscape to release their code illustrated a valuable way to engage with potential software users and

[4] Josh McHugh, *For the Love of Hacking*, Forbes, August 10, 1998.

> developers, and convince them to create and improve source code by participating in an engaged community. The conferees also believed that it would be useful to have a single label that identified this approach and distinguished it from the philosophically- and politically-focused label “free software.” Brainstorming for this new label eventually converged on the term “open source”…[5]

With this new label in hand, OSI presented itself to the world as the arbiter of what constitutes an open source license, and what does not. To this end, it created a list of characteristics that it felt all open source licenses should possess and in 1999 identified fourteen such licenses that met its criteria, including the GPL, the BSD license, the Artistic License (featured in *Jacobsen v. Katzer*, discussed in Section 19.2.4) and the Mozilla Public License. These were the first OSI “certified” licenses. Since then, OSI has slightly expanded its list of characteristics to ten, and has certified over 100 different OSS licenses.[6] OSI’s current list of OSS characteristics is reproduced here.

THE OPEN SOURCE DEFINITION[7]

open source
initiative®

Introduction

Open source doesn’t just mean access to the source code. The distribution terms of open source software must comply with the following criteria:

1. Free Redistribution

The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale.

2. Source Code

The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code,

[5] Open Source Initiative, History of the OSI, https://opensource.org/history. Immediately after the February meeting, Raymond updated *The Cathedral and the Bazaar* to replace the term “free software” with “open source.” www.catb.org/esr/writings/homesteading/cathedral-bazaar/

[6] See https://opensource.org/licenses/alphabetical (as of December 2, 2020, there were 105 different licenses on OSI’s certification list).

[7] https://opensource.org/osd.html.

there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost, preferably downloading via the Internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed.

3. Derived Works

The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

4. Integrity of the Author's Source Code

The license may restrict source-code from being distributed in modified form only if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.

5. No Discrimination Against Persons or Groups

The license must not discriminate against any person or group of persons.

6. No Discrimination Against Fields of Endeavor

The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

7. Distribution of License

The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

8. License Must Not Be Specific to a Product

The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.

9. License Must Not Restrict Other Software

The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open source software.

10. License Must Be Technology-Neutral

No provision of the license may be predicated on any individual technology or style of interface.

Notes and Questions

1. *The OSI definition*. OSI's definition of OSS is clearly inspired by Richard Stallman's ideas, but is phrased in more neutral language. Are there any ways that the OSI definition falls short of Stallman's goals? Does the OSI definition go beyond Stallman's original ideas? Which of the ten OSI attributes of OSS do you think are the most important? The least important?
2. *Free redistribution*. There is considerable confusion in the industry over the ability to charge for OSS. OSI's Definition 1 states that an OSS license "shall not require a royalty or other fee" for the sale or reproduction of OSS software. Yet Richard Stallman himself makes it clear that a software developer may charge for OSS software and emphasizes that the term "free software" "has nothing to do with price." So what does OSI mean in Definition 1?
3. *No copyleft?* OSI's definition is notably silent on the issue of copyleft. Why is this feature of OSS, which was so important to Stallman, omitted from the OSI definition?
4. *Export controls*. OSI Definition 5 requires that OSS not discriminate against any person or group, which sounds like an admirable goal. But OSI explains that Definition 5 is intended to prevent software licensors from prohibiting the export of software to users in countries that are subject to national (i.e., US) export restrictions, such as Cuba, North Korea, Iran and the like. Why would OSI wish to ban prohibitions on such software exports, particularly if they are mandated by law?
5. *Commercial use*. Definition 6 prohibits discrimination against different business models. In particular, it prohibits OSS licenses from containing restrictions on commercial use along the lines of the Creative Commons Non-Commercial (NC) licensing model. OSI goes so far as saying that "We want commercial users to join our community, not feel excluded from it." Why doesn't OSI recognize an "NC" OSS license? Isn't this something that Richard Stallman would approve of?
6. *The anti-NDA clause*. The annotations to the OSI definition explain that Definition 7 is intended to "forbid closing up software by indirect means such as requiring a non-disclosure agreement." Is this how you originally read the definition? What problem is this clause trying to avoid?
7. *Technology neutrality*. OSI Definitions 8 and 10 seek to divorce OSS from any particular software or hardware dependencies. Why is this approach perceived as beneficial?

FIGURE 19.3 Eric Raymond, one of the founders of OSI, in 2004.

8. *OSS license proliferation*. Once OSI set itself up as a certifier of OSS licenses, it received a flood of licenses from groups seeking certification – companies, nonprofit organizations, attorneys, standards bodies and more. As OSI explains,

   This explosion of choice in licensing reflected both the interest in Open Source as well as the many particular ways in which people wanted to create and or manage their Open Source software. Unfortunately, while all of these licenses provide the freedom to read, modify, and share source code, many of the licenses were legally incompatible with other free and open source licenses, seriously constraining the ways in which developers could innovate by combining rather than merely extending Open Source software.[8]

   As a result, in 2004, OSI began a process to "clear out the licensing deadwood so as to make more room (and potentially ensure greater license compatibility) for the more popular licenses." It formed a License Proliferation Committee, which produced a report in 2006 recommending that OSI-certified licenses be classified according to popularity, and that in addition to OSI's substantive criteria for determining whether licenses comply with the OSS definition, certification also take into account three additional questions: (1) Is the license duplicative? (2) Is the license clearly written, simple and understandable? And (3) is the license reusable?[9] Today, there are over 100 OSI-certified OSS licenses, but only 8 in the category "popular and widely-used or with strong communities."[10] These are:

   - Apache License 2.0 (Apache-2.0)
   - 3-clause BSD license (BSD-3-Clause)
   - 2-clause BSD license (BSD-2-Clause)
   - GNU General Public License (GPL)
   - GNU Lesser General Public License (LGPL)
   - MIT license (MIT)
   - Mozilla Public License 2.0 (MPL-2.0)
   - Common Development and Distribution License 1.0 (CDDL-1.0)
   - Eclipse Public License 2.0 (EPL-2.0)

   What is the problem with license proliferation? Why did the License Proliferation Committee recommend that the most popular OSI-certified licenses be identified and grouped together? When might you recommend that a client develop its own OSS license and seek OSI certification for it?

19.2.3 *The BSD Licenses*

Researchers at AT&T Bell Laboratories developed the Unix operating system in the late 1960s and liberally shared its source code with researchers at other institutions. A copy of Unix was sent to the University of California Berkeley in 1974, and in 1978 researchers there released a version of Unix known as the Berkeley Software Distribution, or BSD. Berkeley researchers released their software under various simple licensing terms, and in 1990 standardized their use around what became known as the original BSD license.

[8] OSI, The Licence Proliferation Project, https://opensource.org/proliferation.
[9] OSI, Report of License Proliferation Committee and Draft FAQ (2006), https://opensource.org/proliferation-report.
[10] OSI, Open Source Licenses by Category, https://opensource.org/licenses/category.

FIGURE 19.4 A "daemon" is a type of software agent. This demon in sneakers came to be associated with the BSD project.

The original BSD license contained four short clauses plus a disclaimer of warranties and limitation of liability. One of those clauses (#3) caused considerable consternation in the industry. It read:

3. All advertising materials mentioning features or use of this software must display the following acknowledgement: This product includes software developed by the <organization>.

The problem with the so-called "advertising clause" was its cumulative effect. That is, when a developer used a piece of BSD code distributed by Berkeley, it was not burdensome to include a one-sentence acknowledgment of Berkeley in the ad. But when that developer passed along its software to someone else, who passed it along to someone else, and so on, the number of required acknowledgments quickly outnumbered the actual text of any advertisement. Richard Stallman claims that he counted seventy-five such notices in a 1997 software program released under this license.[11] As a result, Berkeley amended the BSD license in 1999 to remove the advertising clause.

This left a version of the BSD license with three clauses, which became known as the Revised or Modified BSD License. An even simpler version containing just one clause (in addition to the disclaimers) was also released in 1999.

BSD 1-CLAUSE LICENSE (AKA SIMPLIFIED BSD LICENSE)

Copyright (c) [Year]

[Name of Organization] [All rights reserved].

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

[11] Richard Stallman, The BSD License Problem, www.gnu.org/licenses/bsd.html

THIS SOFTWARE IS PROVIDED BY [Name of Organization] "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL [Name of Organization] BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

BSD 3-CLAUSE LICENSE (AKA REVISED OR MODIFIED BSD LICENSE)

Copyright <YEAR> <COPYRIGHT HOLDER>

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The BSD licenses were among the first to be certified by OSI. Today, the BSD licenses are widely used, largely because of their simplicity and their lack of restrictions and obligations. They contain no copyleft or other burdensome restrictions or obligations. To use an analogy from the world of real property conveyances, the BSD licenses most closely resemble quitclaim deeds – they allow any use of the licensed software in source and object code forms, and release the provider from all liability. As we will see below, this simplified approach diverged significantly from that of Richard Stallman and the Free Software Foundation (FSF).

Notes and Questions

1. *BSD and OSI compliance?* OSI lists ten elements that define OSS licenses. The BSD licenses were among the first that it certified in 1999. How do the BSD licenses embody the ten OSI definitional elements?
2. *Attribution.* As discussed in Section 19.1 (Note 4), creators of copyrighted works are most interested in getting credit for their work, even if they give it away for free. The BSD and most other OSS licenses (including the GNU GPL) provide for attribution by requiring that subsequent distributors of OSS software reproduce any copyright notices that are included in the original source code. The theory is that when a user modifies a portion of that code, it should add itself to the copyright notice, thereby accumulating a list of all contributors to the code. Is this a sensible approach to attribution? Why don't OSS licenses use the simpler approach exemplified by the CC BY licensing tag?
3. *Corporate appropriation?* The BSD licenses contain no copyleft requirement, meaning that a recipient of BSD-licensed code can take that code, modify it and include it in a proprietary software program distributed under a traditional, non-OSS license. In other words, code that was once OSS can be appropriated and turned into proprietary code. Not surprisingly, the FSF and other OSS advocates objected strongly to this possibility. But it has made the BSD license extremely popular among corporate users of OSS. Do you think that the original BSD license drafters at UC Berkeley were right to omit a copyleft provision or not? How does the historical development of BSD (as opposed to the GNU project) help to explain why this approach was chosen?
4. *Disclaimers.* The BSD licenses are famously (and refreshingly) short. In fact, the majority of their text is devoted to a disclaimer of warranties and limitation of liability. Why are liability disclaimers the focus of these agreements? How do these liability provisions differ between the 1-Clause and 3-Clause BSD licenses?

19.2.4 *The GNU General Public License*

Without a doubt, the most famous and infamous OSS license is the Free Software Foundation's (FSF) GNU General Public License (GPL). Richard Stallman released version 1 of the GPL in February 1989. The agreement embodied the principles of freedom and copyleft that he espoused when creating the GNU project. The original GPL was short by the standards of IP licensing agreements, running to just over 1,600 words. Version 2, which added some corrections and clarifications, was released in 1991 and amounted to around 2,500 words. GPL v2 was adopted broadly by many significant OSS projects, most notably the Linux operating system. And even though Stallman was not part of the OSI project, and in fact vocally objected to its rejection of his "free software" terminology, the GPL heavily influenced OSI's definition of OSS and GPL v2 was the first OSS license to be certified by OSI.

But over the years, as the law and norms of the software industry evolved, GPL v2 started to become outdated. Concerns emerged over how the GPL should handle developments such as the Digital Millennium Copyright Act (DMCA) of 1998, digital rights management, software patents and compatibility with the dozens of new OSS licenses being certified by OSI. As a result, in 2005 the FSF began a public consultation process to update the GPL. Over the eighteen-month consultation period, more than 2,600 written comments were submitted. GPL v3 was released in June 2007, comprising over 5,200 words.

GPL v3 was controversial for a number of reasons, including its treatment of patents and digital rights management. As a result, many OSS projects, including Linux, declined to "upgrade" from GPL v2 to v3 (more on this below). Nevertheless, the GPL licenses, principally v2 and v3, remain important documents in the OSS ecosystem. Their language is, however, notoriously turgid and requires a significant amount of background knowledge and lore to parse. For this reason, it is not reproduced here. Instead, some of its more significant and controversial provisions are summarized and discussed below.[12]

19.2.4.1 Access to Source Code

The *sine qua non* of OSS licensing is the availability of the source code underlying a computer program. The GPL, which was created with the goal of source code availability in mind, contains detailed prescriptions on when and how source code must be provided or made available to recipients of the software. Importantly, the requirement to deliver source code can be met in a variety of ways, including by providing a physical disc or other medium, making it available for download from a network server or peer-to-peer service or, if the software is embedded in a physical device, extending a written offer valid for three years to provide such source code.

19.2.4.2 Copyleft: The "Viral" Nature of GPL

Today, the GPL is probably best known for the "viral" effect of its copyleft provisions. That is, if a piece of software is distributed under the GPL, then anyone who redistributes that software, *or any modified version of that software*, must also distribute it under the GPL. Thus, like a biological virus, the GPL propagates itself from user to user. But the real threat perceived by the GPL was not the continuing need to license GPL'd code under the GPL, but the risk that the GPL'd code could infect any proprietary code with which it was combined, making the entire combined work subject to the GPL.

To be specific, § 5(c) of the GPL provides that "You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy." Section 0 defines a "covered work" as "either the unmodified Program or a work based on the Program," and provides that to

[12] The interested student may find the full text of GPL v3 (and prior versions of the GPL) at www.gnu.org/licenses/gpl-3.0.en.html.

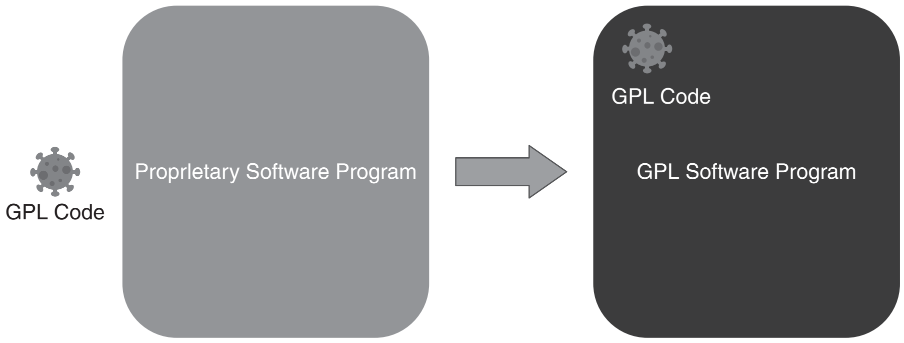


FIGURE 19.5 Graphical illustration of the perceived "viral effect" of GPL software combined with proprietary software. OSS advocates claim that representations like this overstate the risk of using GPL software.

"modify" a work means "to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a 'modified version' of the earlier work or a work 'based on' the earlier work." While these provisions, read together, are less than clear, they are generally understood to mean that a larger program that incorporates GPL'd software should itself become subject to the GPL.

This being said, the GPL does recognize an exception of "mere aggregation" of separate works on the same "storage or distribution medium." Thus, distributing a proprietary program and a GPL program on the same CD would not "infect" the proprietary program with the GPL. This exception, however, has given cold comfort to commercial users, who are generally more concerned with proprietary programs that might actually call or access the GPL'd code.

19.2.4.3 Anti-Anti-Circumvention

The enactment of the DMCA in 1998 galled many in the OSS and hacker communities, particularly the "anti-circumvention" provisions of 17 U.S.C. § 1201. These provisions made it illegal to attempt to circumvent any "technological measure that effectively controls access to a [copyrighted] work." As such, the DMCA prohibited the hacking of encryption and digital rights management protections. In response, GPL v3 expressly states that any software code covered by the GPL will *not* be deemed to be part of such a technological measure. In other words, it ensured that no one would be liable for hacking any code covered by the GPL, whether it was used in a protection system or not.

19.2.4.4 Anti-Tivoization

In 1999, TiVo released one of the first consumer digital video recorders (DVR), which allowed viewers to make digital recordings of broadcast television programs. As a significant improvement over tape-based VCR machines, the TiVo DVR became incredibly popular. It incorporated the Linux kernel, which was licensed under GPL v2. But because the Linux software controlled a complex hardware device, TiVo prevented users from uploading modified versions of the Linux software to the DVR. This restriction infuriated the FSF and Richard Stallman, who claimed that TiVo had violated the GPL in spirit, if not in fact. So when GPL v3 was proposed in 2005, it contained a provision that became known as the "Anti-Tivoization" clause. The lengthy clause, which is included in Section 6 of GPL v3, was heavily negotiated and bears

FIGURE 19.6 In 1999 TiVo introduced the first successful mass-market DVR device. It ran the Linux kernel.

the signs of a Frankenstein contractual clause negotiated by committee. In effect, it requires that consumer hardware devices intended for home use (i.e., not equipment used in hospitals, factories, etc.) allow end users to upload modified versions of GPL software, so long as this will not interfere with their operation. It also permits the manufacturer to void warranties and service commitments when modified software is installed.

There were significant objections to the Anti-Tivoization clause, including by Linus Torvalds, the creator of Linux. Torvalds believed that hardware manufacturers were entitled to prevent users from uploading modified software to their hardware products, and didn't see why doing so violated the OSS spirit. As a result, Torvalds never "upgraded" the Linux license from GPL v2 to v3 (and Linux remains under v2 today).

19.2.4.5 Patentleft

The preface to the GPL states that "every program is threatened constantly by software patents." But concerns about patents and OSS are not unique to the FSF and OSS advocates. Professor Greg Vetter points out that patent law may be "particularly threatening" to [OSS] for a variety of reasons.[13] And IBM, the holder of one of the largest patent portfolios in the world, observed in 2005 that

> Patents … must be considered when OSS is developed. When OSS is created and licensed, it must, as a practical matter, carry with it a grant of license to any patents concerning the software that the author holds. Doing otherwise creates an untenable situation, wherein any users of that OSS may become inadvertent infringers of the patent. Some licenses … include an explicit grant of a patent license, but most do not.[14]

To address patent issues, "the GPL assures that patents cannot be used to render [a] program non-free." How, exactly, does the GPL do this? In typical fashion, the answer is complex. It involves four parts, all contained in § 11 of the GPL:

i. *Present and Future Patent License*. Section 11, ¶ 2 describes "essential patent claims" as all patent claims owned or controlled by a contributor to the licensed code currently or in the future. Under ¶ 3, each such contributor grants to each user of the code

[13] Greg R. Vetter, *Commercial Free and Open Source Software: Knowledge Production, Hybrid Appropriability, and Patents*, 77 Fordham L. Rev. 2087, 2093 (2009).

[14] Peter G. Capek, et al., *A History of IBM's Open-Source Involvement and Strategy*, 44 IBM Syst. J. 249 (2005).

> a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

The GPL thus makes it clear that a patent holder who distributes software under the GPL ("Licensor") cannot later sue users of that software for patent infringement.

One concern with this provision is that it covers not only the software contribution made by the licensor, but all other software contained in the relevant GPL program. Thus, if a licensor obtains a GPL program to which seventy-five previous contributors have contributed, then modifies it and redistributes it under the GPL, the licensor's patents are licensed with respect to the entire GPL program, even the portions written by the other seventy-five contributors. This is the case even if some of those other contributors were intentionally infringing the licensor's patents.[15]

ii. *Third Party Licenses*. In addition to the patent license described above, Section 11, ¶ 5 addresses patents held by third parties. It provides that:

> If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients.

Thus, if a licensor of GPL code has the benefit of a license under a third-party patent, and recipients of that GPL code do not have the right to operate under that third-party patent license (i.e., the licensor does not have the right to sublicense), then the licensor must do one of three things. The first and third options are effectively the same: the licensor must extend the rights under the patent license to all users of the GPL code on a royalty-free basis (i.e., "under the terms of this License"). In most cases, this will be impossible. This leaves the licensor with option 2, under which it must disavow the benefits of the patent license with respect to itself. How it would do this is unclear, but the idea, presumably, is that the licensor will be motivated to find ways to extend patent licenses or sublicenses to users of the GPL code if it is itself stripped of the benefit of its patent licenses.

iii. *No One-Off Patent Licenses*. Section 11, ¶ 6 addresses a situation in which a licensor grants a patent license to some (but not all) parties receiving a piece of GPL code (e.g., in a litigation settlement). If that happens, then the license "is automatically extended to all recipients of the covered work and works based on it." Again, it is not clear how this automatic expansion of rights would legally occur, but it will certainly make licensors think twice before granting one-off patent licensors to recipients of GPL code.

[15] The GPL is not unique in requiring a broad patent license covering an entire product, as opposed to the licensor's contributions. The Mozilla Public License takes a similar approach. In contrast, the Apache 2.0 license contains a patent license that is limited to the licensor's contributions:

> each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted.

iv. *Discriminatory Licenses*. Section 11, ¶ 7 seeks to eliminate the benefits that Licensors and selected users might receive from cross-licenses and other private arrangements. As one commentator explains:

> Section 11(7) stemmed from a 2006 cross-licensing agreement between Microsoft and Novell. As a result of this agreement, Microsoft and Novell customers were granted protection against the other party's patent claims. Section 11(7) was incorporated into the draft licence text at the time; it is a narrow clause which is tailored at such cross-licensing agreements. It covers only one of numerous possible cases which can be easily circumvented by minor modifications made by the affected entities.[16]

Notes and Questions

1. *The SaaS loophole and the Affero General Public License*. The GPL requires that anyone who "conveys" software covered by the GPL to another must provide the recipient with the source code of that software. But the GPL contains an important exception stating that "Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying." That is, an entity's use of GPL-licensed software *to provide services to others* does not constitute a conveyance, so long as software code is not actually transferred to the service recipient. This "loophole" allows firms to obtain software under the GPL, modify its source code, then make that modified software remotely available to users on a "software as a service" (SaaS) basis (see Section 18.3), all while avoiding the obligation to make their own source code modifications publicly available. The appearance of this "SaaS loophole" in GPL v3 (also known as the "ASP" [application service provider] loophole) shocked many in the OSS community, and seemed to contradict the fundamental "open source" precepts of the FSF.[17] Yet, through the difficult negotiation and public commenting process that led to GPL v3, it remained. The concession to OSS purists was the concurrent release in 2007 of the GNU Affero General Public License, a version of the GPL that closes the SaaS loophole by providing that

> if you modify the Program, your modified version must prominently offer all users interacting with it remotely through a computer network (if your version supports such interaction) an opportunity to receive the Corresponding Source of your version by providing access to the Corresponding Source from a network server at no charge, through some standard or customary means of facilitating copying of software.

The degree to which the Affero GPL has been adopted is uncertain. Some commentators see evidence that it is gaining in popularity.[18] Yet major online service providers such as Google have reportedly banned its use.[19] How would you advise a client that is interested in offering a SaaS environment using OSS? Would you recommend avoiding or embracing the Affero GPL?

[16] Hendrick Schöttle, *Open Source Software and Patents: How the GPLv3 Affects Patent Portfolios*, Intl. L. Off., February 5, 2013, www.internationallawoffice.com/Newsletters/Tech-Data-Telecoms-Media/International/Osborne-Clarke/Open-source-software-and-patents-how-the-GPLv3-affects-patent-portfolios.

[17] Richard Stallman has written at length about the nonfree nature of SaaS services. Richard Stallman, *Who Does That Server Really Serve?*, www.gnu.org/philosophy/who-does-that-server-really-serve.en.html.

[18] Phil Odence, *The Quietly Accelerating Adoption of the AGPL*, August 14, 2017, www.synopsys.com/blogs/software-security/using-agpl-adoption.

[19] Cade Metz, *Google Open Source Guru: "Why We Ban the AGPL,"* March 31, 2011, www.theregister.com/2011/03/31/google_on_open_source_licenses.

2. *The LGPL*. When the FSF released GPL v2 in 1991, it also released a licensing agreement called the "Library GPL" or LGPL. The LGPL was intended to be used with software "libraries" – standalone software modules used to perform discrete functions, such as time zone conversions or the calculation of square roots. Libraries can be used by application programs such as databases, spreadsheets and word processors, but remain relatively independent of these larger programs.

   In the early days, commercial software developers were reluctant to use libraries released under the GPL because they were concerned that "linking" a commercial program to the library would cause the entire linked body of software (application plus library) to become GPL software (i.e., through the viral effect of the GPL's copyleft provisions). To address this concern, the FSF developed the LGPL.

   For most purposes, a software library released under the LGPL is treated just like software released under the ordinary GPL. The source code of the library is provided to users, and if it is modified, the modified source code is also covered by the LGPL. The major difference between the LGPL and the ordinary GPL is how they treat other programs that are "linked" to the covered software. Section 5 of the original LGPL provided that

   > A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.[20]

   Thus, if I create a time zone utility and release it as a library under the LGPL, and UPS wishes to use this utility in its proprietary delivery scheduling software, the LGPL library will not "contaminate" UPS's proprietary software so long as my library is kept separate from the proprietary software.

   This exception to the copyleft feature of the GPL was enthusiastically welcomed by corporate software developers. Now they could use OSS libraries without the risk of contaminating their proprietary software. But it was perhaps this very enthusiasm that caused the FSF to step back from the LGPL licensing model. When it updated the LGPL in 1999, it changed its name from the "Library GPL" to the "*Lesser* GPL" and published a warning to developers titled "Why you shouldn't use the Lesser GPL for your next library."[21]

   Given the importance of distinguishing between a work that "uses" an LGPL library and a work "based on" an LGPL library, a significant amount of lore and guidance has developed over the years, most of which is comprehensible only to computer programmers (and then only partially). For example, much of the debate focuses on whether a proprietary software program links with an LGPL library in a manner that is "static" (embedding the library into the code of the proprietary program) or "dynamic" (where a proprietary program accesses the library "on the fly" as it is executed). Many commentators argue that dynamic linking should not result in a combined program "based on" the library, while static linking *could* result in a combined program subject to the copyleft terms of the LGPL. But the FSF itself seems to take the position that *both* static and dynamic linking create such a combined program.[22] While the issue has never been litigated, many companies continue to view the LGPL as a relatively "safe" OSS license.

[20] Later versions of the LGPL have altered (and obfuscated) this text substantially.

[21] Free Software Fndn, *Why You Shouldn't Use the Lesser GPL for Your Next Library*, 1999, www.gnu.org/licenses/why-not-lgpl.html.

[22] See Michael Pavento, *A Practical Guide to Open Source Software* 3–4 (2012) (the linking debate).

Why do you think the FSF discourages use of its own LGPL? If the FSF dislikes the LGPL so much, why didn't it revoke the license entirely, instead of reissuing it with a warning? As an attorney advising a software client, how would you explain the LGPL and its potential effect when the OSS community and the FSF themselves cannot seem to agree on its precise meaning?

3. *The GPL and patents*. GPL v3 contains more patent-related language than any other OSS license and seeks rights well beyond the straightforward, though broad, patent license contained in § 11, ¶ 3. Why are the additional rights under ¶¶ 5, 6 and 7 needed? What would be the effect of eliminating these provisions from the GPL?
4. *Implied patent licenses*. Unlike the GPL, the BSD licenses contain no provisions relating to patents. Is it likely that someone distributing software under the BSD license could later sue users for patent infringement? GPL v2 likewise contains no explicit patent license, but commentators have suggested that users of GPL v2 code would have a strong implied license argument if the distributor later sued them for patent infringement.[23] And § 11, ¶ 8 of the GPL v3 itself states that nothing in the GPL "shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law." Does a user of OSS software licensed under these different licenses have an implied license under the patents of the OSS licensor? If so, what is the likely scope of this implied license? Why do you think the FSF included so much language about patents in GPL v3 if implied licenses were already understood to exist under OSS programs, including under GPL v2? (See Chapter 4 for a discussion of implied licenses.)
5. *Microsoft and the Linux patent litigation*. The FSF was somewhat vindicated in its worries about patents when, in 2009, Microsoft began to file patent infringement suits, first against electronics manufacturers using the Linux kernel, and later against mobile device makers using the Android operating system. According to one estimate, Microsoft earned $3.4 billion from patent licenses on Android, including $1 billion from Samsung alone.[24] As noted above, Linux and its variants (including Android) are licensed under GPL v2, which lacks the patent clauses of GPL v3. But Microsoft, which was not a major contributor to these OSS projects, would not likely have been subject to the patent licensing provisions of GPL v3 even if they had applied. What, if anything, could the FSF have done to prevent Microsoft's litigation campaign?[25]
6. *Defensive termination*. The Apache License 2.0, which grants a patent license more limited in scope than that under GPL v3, § 11, ¶ 3, contains a provision that is not found in the GPL. Section 3 of the Apache License provides that

   If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

[23] Pavento, *supra* note 21, at 8.

[24] See Steven J. Vaughan-Nichols, *Microsoft Open-Sources Its Patent Portfolio*, ZDNet, October 10, 2018, www.zdnet.com/article/microsoft-open-sources-its-entire-patent-portfolio.

[25] In 2018, Microsoft reversed course and joined the Open Innovation Platform, pledging to allow rivals to operate under 60,000 Microsoft patents relating to Linux without charge. Klint Finley, *Microsoft Calls a Truce in the Linux Patent Wars*, Wired, November 10, 2018.

This is called a "defensive termination" clause. It results in the automatic termination of the patent licenses granted by the OSS licensor if the user sues the licensor for patent infringement with respect to the licensed OSS. Why do you think such a clause is not included in the GPL? (See Section 20.1.4 for a discussion of defensive termination in the context of licenses of standards-essential patents.) Note that copyright licenses are not terminated by this provision. Why not?

19.2.5 *Enforcement of OSS Licenses*

In the discussion of OSS licenses to this point, we have assumed that such licenses are binding and enforceable legal agreements. Yet this was not always clear. There have been a handful of cases in the United States and Europe interpreting and enforcing the terms of OSS licenses.[26] The following is among the best known of these.

Jacobsen v. Katzer

535 F.3d 1373 (Fed. Cir. 2008)

HOCHBERG, DISTRICT JUDGE (by designation)

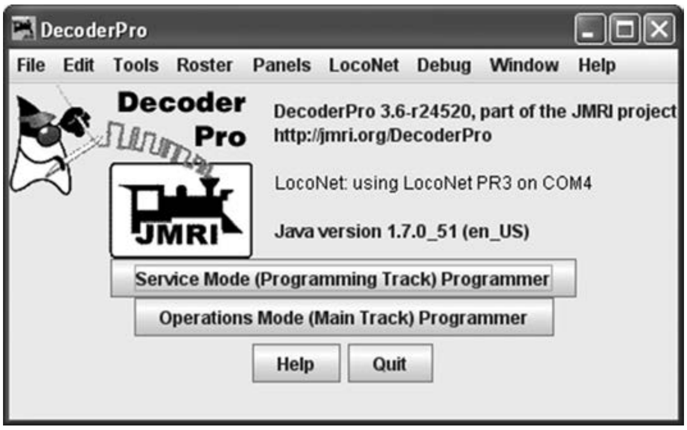
We consider here the ability of a copyright holder to dedicate certain work to free public use and yet enforce an "open source" copyright license to control the future distribution and modification of that work. Appellant Robert Jacobsen ("Jacobsen") appeals from an order denying a motion for preliminary injunction. Jacobsen holds a copyright to computer programming code. He makes that code available for public download from a website without a financial fee pursuant to the Artistic License, an "open source" or public license. Appellees Matthew Katzer and Kamind Associates, Inc. (collectively "Katzer/Kamind") develop commercial software products for the model train industry and hobbyists. Jacobsen accused Katzer/Kamind of copying certain materials from Jacobsen's website and incorporating them into one of Katzer/Kamind's software packages without following the terms of the Artistic License. Jacobsen brought an action for copyright infringement and moved for a preliminary injunction.

The District Court held that the open source Artistic License created an "intentionally broad" nonexclusive license which was unlimited in scope and thus did not create liability for copyright infringement [and] denied the motion for a preliminary injunction. We vacate and remand.

I

Jacobsen manages an open source software group called Java Model Railroad Interface ("JMRI"). Through the collective work of many participants, JMRI created a computer programming application called DecoderPro, which allows model railroad enthusiasts to use their computers to program the decoder chips that control model trains. DecoderPro files are available for download and use by the public free of charge from an open source incubator website called SourceForge; Jacobsen maintains the JMRI site on SourceForge.

[26] For an only slightly outdated compendium of cases, see Heather J. Meeker, *Open Source and the Age of Enforcement*, 4 Hastings Sci. & Tech. L.J. 267 (2012).


DecoderPro
File Edit Tools Roster Panels LocoNet Debug Window Help
Decoder Pro
JMRI
Service Mode (Programming Track) Programmer
Operations Mode (Main Track) Programmer
Help
Quit

DecoderPro 3.6-r24520, part of the JMRI project
http://jmri.org/DecoderPro

LocoNet: using LocoNet PR3 on COM4

Java version 1.7.0_51 (en_US)

FIGURE 19.7 Screenshot from the DecoderPro model railroad control software released by Jacobsen for the JMRI project.

The downloadable files contain copyright notices and refer the user to a "COPYING" file, which clearly sets forth the terms of the Artistic License.

Katzer/Kamind offers a competing software product, Decoder Commander, which is also used to program decoder chips. During development of Decoder Commander, one of Katzer/Kamind's predecessors or employees is alleged to have downloaded the decoder definition files from DecoderPro and used portions of these files as part of the Decoder Commander software. The Decoder Commander software files that used DecoderPro definition files did not comply with the terms of the Artistic License. Specifically, the Decoder Commander software did not include (1) the authors' names, (2) JMRI copyright notices, (3) references to the COPYING file, (4) an identification of SourceForge or JMRI as the original source of the definition files, and (5) a description of how the files or computer code had been changed from the original source code. The Decoder Commander software also changed various computer file names of DecoderPro files without providing a reference to the original JMRI files or information on where to get the Standard Version.

Jacobsen moved for a preliminary injunction, arguing that the violation of the terms of the Artistic License constituted copyright infringement and that, under Ninth Circuit law, irreparable harm could be presumed in a copyright infringement case. The District Court found that Jacobsen had a cause of action only for breach of contract, rather than an action for copyright infringement based on a breach of the conditions of the Artistic License. Because a breach of contract creates no presumption of irreparable harm, the District Court denied the motion for a preliminary injunction.

Jacobsen appeals the finding that he does not have a cause of action for copyright infringement. Although an appeal concerning copyright law and not patent law is rare in our Circuit, here we indeed possess appellate jurisdiction. In the district court, Jacobsen's operative complaint against Katzer/Kamind included not only his claim for copyright infringement, but also claims seeking a declaratory judgment that a patent issued to Katzer is not infringed by Jacobsen and is invalid. Therefore the complaint arose in part under the patent laws.

II. A

Public licenses, often referred to as "open source" licenses, are used by artists, authors, educators, software developers, and scientists who wish to create collaborative projects and to dedicate certain works to the public. Several types of public licenses have been designed to provide creators of copyrighted materials a means to protect and control their copyrights. Creative Commons, one of the amici curiae, provides free copyright licenses to allow parties to dedicate their works to the public or to license certain uses of their works while keeping some rights reserved.

Open source licensing has become a widely used method of creative collaboration that serves to advance the arts and sciences in a manner and at a pace that few could have imagined just a few decades ago. For example, the Massachusetts Institute of Technology ("MIT") uses a Creative Commons public license for an OpenCourseWare project that licenses all 1800 MIT courses. Other public licenses support the GNU/Linux operating system, the Perl programming language, the Apache web server programs, the Firefox web browser, and a collaborative web-based encyclopedia called Wikipedia. Creative Commons notes that, by some estimates, there are close to 100,000,000 works licensed under various Creative Commons licenses. The Wikimedia Foundation, another of the amici curiae, estimates that the Wikipedia website has more than 75,000 active contributors working on some 9,000,000 articles in more than 250 languages.

Open Source software projects invite computer programmers from around the world to view software code and make changes and improvements to it. Through such collaboration, software programs can often be written and debugged faster and at lower cost than if the copyright holder were required to do all of the work independently. In exchange and in consideration for this collaborative work, the copyright holder permits users to copy, modify and distribute the software code subject to conditions that serve to protect downstream users and to keep the code accessible. By requiring that users copy and restate the license and attribution information, a copyright holder can ensure that recipients of the redistributed computer code know the identity of the owner as well as the scope of the license granted by the original owner. The Artistic License in this case also requires that changes to the computer code be tracked so that downstream users know what part of the computer code is the original code created by the copyright holder and what part has been newly added or altered by another collaborator.

Traditionally, copyright owners sold their copyrighted material in exchange for money. The lack of money changing hands in open source licensing should not be presumed to mean that there is no economic consideration, however. There are substantial benefits, including economic benefits, to the creation and distribution of copyrighted works under public licenses that range far beyond traditional license royalties. For example, program creators may generate market share for their programs by providing certain components free of charge. Similarly, a programmer or company may increase its national or international reputation by incubating open source projects. Improvement to a product can come rapidly and free of charge from an expert not even known to the copyright holder. The Eleventh Circuit has recognized the economic motives inherent in public licenses, even where profit is not immediate. See *Planetary Motion, Inc. v. Techsplosion, Inc.*, 261 F.3d 1188, 1200 (11th Cir. 2001) (Program creator "derived value from the distribution [under a public license] because he was able to improve his Software based on suggestions sent by end-users … It is logical that as the Software improved, more end-users used his Software, thereby increasing [the programmer's] recognition in his profession and the likelihood that the Software would be improved even further.").

B

The parties do not dispute that Jacobsen is the holder of a copyright for certain materials distributed through his website. Katzer/Kamind also admits that portions of the DecoderPro software were copied, modified, and distributed as part of the Decoder Commander software. Accordingly, Jacobsen has made out a prima facie case of copyright infringement. Katzer/Kamind argues that they cannot be liable for copyright infringement because they had a license to use the material. Thus, the Court must evaluate whether the use by Katzer/Kamind was outside the scope of the license. The copyrighted materials in this case are downloadable by any user and are labeled to include a copyright notification and a COPYING file that includes the text of the Artistic License. The Artistic License grants users the right to copy, modify, and distribute the software:

> provided that [the user] insert a prominent notice in each changed file stating how and when [the user] changed that file, and provided that [the user] do at least ONE of the following:

a) place [the user's] modifications in the Public Domain or otherwise make them Freely Available, such as by posting said modifications to Usenet or an equivalent medium, or placing the modifications on a major archive site such as ftp.uu.net, or by allowing the Copyright Holder to include [the user's] modifications in the Standard Version of the Package.
b) use the modified Package only within [the user's] corporation or organization.
c) rename any non-standard executables so the names do not conflict with the standard executables, which must also be provided, and provide a separate manual page for each nonstandard executable that clearly documents how it differs from the Standard Version, or
d) make other distribution arrangements with the Copyright Holder.

The heart of the argument on appeal concerns whether the terms of the Artistic License are conditions of, or merely covenants to, the copyright license. Generally, a "copyright owner who grants a nonexclusive license to use his copyrighted material waives his right to sue the licensee for copyright infringement" and can sue only for breach of contract. If, however, a license is limited in scope and the licensee acts outside the scope, the licensor can bring an action for copyright infringement.

Thus, if the terms of the Artistic License allegedly violated are both covenants and conditions, they may serve to limit the scope of the license and are governed by copyright law. If they are merely covenants, by contrast, they are governed by contract law. The District Court did not expressly state whether the limitations in the Artistic License are independent covenants or, rather, conditions to the scope; its analysis, however, clearly treated the license limitations as contractual covenants rather than conditions of the copyright license.

Jacobsen argues that the terms of the Artistic License define the scope of the license and that any use outside of these restrictions is copyright infringement. Katzer/Kamind argues that these terms do not limit the scope of the license and are merely covenants providing contractual terms for the use of the materials, and that his violation of them is neither compensable in damages nor subject to injunctive relief. Katzer/Kamind's argument is premised upon the assumption that Jacobsen's copyright gave him no economic rights because he made his computer code available to the public at no charge. From this assumption, Katzer/Kamind argues that copyright law does not recognize a cause of action for non-economic rights. The District Court based its opinion on the breadth of the Artistic License terms, to which we now turn.

III

The Artistic License states on its face that the document creates conditions: "The intent of this document is to state the conditions under which a Package may be copied." The Artistic License also uses the traditional language of conditions by noting that the rights to copy, modify, and distribute are granted "provided that" the conditions are met. Under California contract law, "provided that" typically denotes a condition.

The conditions set forth in the Artistic License are vital to enable the copyright holder to retain the ability to benefit from the work of downstream users. By requiring that users who modify or distribute the copyrighted material retain the reference to the original source files, downstream users are directed to Jacobsen's website. Thus, downstream users know about the collaborative effort to improve and expand the SourceForge project once they learn of the "upstream" project from a "downstream" distribution, and they may join in that effort.

The District Court interpreted the Artistic License to permit a user to "modify the material in any way" and did not find that any of the "provided that" limitations in the Artistic License served to limit this grant. The District Court's interpretation of the conditions of the Artistic License does not credit the explicit restrictions in the license that govern a downloader's right to modify and distribute the copyrighted work. The copyright holder here expressly stated the terms upon which the right to modify and distribute the material depended and invited direct contact if a downloader wished to negotiate other terms. These restrictions were both clear and necessary to accomplish the objectives of the open source licensing collaboration, including economic benefit. Moreover, the District Court did not address the other restrictions of the license, such as the requirement that all modification from the original be clearly shown with a new name and a separate page for any such modification that shows how it differs from the original.

Copyright holders who engage in open source licensing have the right to control the modification and distribution of copyrighted material. As the Second Circuit explained in *Gilliam v. ABC*, 538 F.2d 14, 21 (2d Cir. 1976), the "unauthorized editing of the underlying work, if proven, would constitute an infringement of the copyright in that work similar to any other use of a work that exceeded the license granted by the proprietor of the copyright." Copyright licenses are designed to support the right to exclude; money damages alone do not support or enforce that right. The choice to exact consideration in the form of compliance with the open source requirements of disclosure and explanation of changes, rather than as a dollar-denominated fee, is entitled to no less legal recognition. Indeed, because a calculation of damages is inherently speculative, these types of license restrictions might well be rendered meaningless absent the ability to enforce through injunctive relief.

In this case, a user who downloads the JMRI copyrighted materials is authorized to make modifications and to distribute the materials "provided that" the user follows the restrictive terms of the Artistic License. A copyright holder can grant the right to make certain modifications, yet retain his right to prevent other modifications. Indeed, such a goal is exactly the purpose of adding conditions to a license grant. The Artistic License, like many other common copyright licenses, requires that any copies that are distributed contain the copyright notices and the COPYING file.

It is outside the scope of the Artistic License to modify and distribute the copyrighted materials without copyright notices and a tracking of modifications from the original computer files. If a downloader does not assent to these conditions stated in the COPYING file, he is instructed to "make other arrangements with the Copyright Holder." Katzer/Kamind did not make any such "other arrangements." The clear language of the Artistic License creates conditions to protect the economic rights at issue in the granting of a public license. These conditions govern the rights to modify and distribute the computer programs and files included in the downloadable software package. The attribution and modification transparency requirements directly serve to drive traffic to the open source incubation page and to inform downstream users of the project, which is a significant economic goal of the copyright holder that the law will enforce. Through this controlled spread of information, the copyright holder gains creative collaborators to the open source project; by requiring that changes made by downstream users be visible to the copyright holder and others, the copyright holder learns about the uses for his software and gains others' knowledge that can be used to advance future software releases.

IV

For the aforementioned reasons, we vacate and remand. While Katzer/Kamind appears to have conceded that they did not comply with the aforedescribed conditions of the Artistic License, the District Court did not make factual findings on the likelihood of success on the merits in proving that Katzer/Kamind violated the conditions of the Artistic License.

The judgment of the District Court is vacated and the case is remanded for further proceedings consistent with this opinion.

FIGURE 19.8 *Jacobsen v. Katzer* concerned OSS used to control model trains.

Notes and Questions

1. *The artistic license*. The OSS license used by Jacobsen was the Artistic License, an OSI-certified, but relatively uncommon, license (OSI places the Artistic License in the "Other/Miscellaneous" category). Given this, how useful is *Jacobsen v. Katzer* for interpreting other, more popular, OSS licenses, such as the GPL and BSD licenses? Does the court's holding extend generally to all OSS licenses, or is it specific to the Artistic License?
2. *Economic harm and free software*. Katzer argued that Jacobsen was not entitled to any economic damages "because he made his computer code available to the public at no charge." What did the court think of this argument? Assuming that Katzer breached the attribution requirement of the Artistic License, what harm did Jacobsen suffer?
3. *Covenant versus condition*. Did the court find that the attribution requirements of the Artistic License were contractual covenants or conditions to the copyright license? What is the significance of this distinction?
4. *What breach*? The district court in *Jacobsen* did not make factual findings regarding the scope of Katzer's use of the DecoderPro software. On remand, what did Katzer need to show to avoid liability?
5. *What dispute*? The district court in *Jacobsen* notes that "[Katzer and Kamind] represent that they have voluntarily ceased all potentially infringing activities utilizing any of the disputed material and … both parties conceded that the disputed material is no longer of value." If this is the case, why did the parties continue to litigate? What did Jacobsen hope to gain with the injunction that he sought?[27]

Problem 19.1

Softbot downloads a copy of the PlanEt workflow planning software from Mikro Software, Inc. (MSI) for $100. In addition, Softbot pays MSI $5,000 for a copy of the software source code and agrees to a one-year maintenance agreement with MSI. The PlanEt source code is licensed under GPL v3, and MSI is listed as its owner in the copyright notice, along with two of its employees. Softbot incorporates the PlanEt code into its Factotum factory management system and begins distributing it to large manufacturing entities around the world. The price of Factotum is $3 million. Before incorporating the PlanEt code into Factotum, Softbot makes significant modifications. When Softbot distributes Factotum, it requires the customer to sign a customary software licensing agreement that prohibits reverse engineering, accessing the source code and attempting to modify the code. What legal recourse does MSI have against Softbot?

19.3 OPEN SOURCE IN THE COMMERCIAL MARKET

Early OSS advocates like Richard Stallman and the FSF felt that OSS should never be combined with proprietary, commercial software. As Stallman famously wrote in 1999, "To permit such combinations would open a hole big enough to sink a ship." This anti-corporate sentiment, and the "viral" nature of Stallman's GPL, frightened corporate IT managers and software developers. They feared that using even a tiny piece of GPL code in a commercial program

[27] See Meeker, supra note 25, at 277 (on remand, the district court issued an injunction in favor of Jacobsen. The case settled soon thereafter).

FIGURE 19.9 Major OSS successes include the Linux and Android operating systems, the Apache web server, the Firefox browser and Red Hat, which provides services related to Linux.

could result in the entire program becoming OSS – a potentially catastrophic result for a company in the business of selling proprietary software.

Yet as the market for OSS grew, and OSS products like Linux, the Apache web server and the Android mobile operating system began to be adopted globally, that fear began to diminish. Nonviral OSS licenses such as BSD, MIT and Apache were viewed as "friendly" to proprietary software. And even GPL code such as Linux could be used safely within a corporate enterprise or in a commercial system, so long as modifications were not made to the software itself, and it was well-segregated from any proprietary programs with which it was distributed.

Today, OSS has come far from its underground, countercultural origins in the 1970s and has assumed a prominent place in the mainstream software industry. In 2019 IBM paid $34 billion for RedHat, a pioneer in distributing and providing services for Linux and other OSS tools, and in 2018 Microsoft paid $7.5 billion in stock to acquire GitHub, a leading platform for OSS development. Deals of this magnitude signal that major corporations view OSS "not as a fad or an adjunct but as a core part of how [they] will make software in the future."[28] In this section we explore how OSS is integrated into commercial software products, services and business models.

19.3.1 *Open Source as a Business Model*

What is the thinking behind corporate strategies involving OSS? The following two excerpts present different perspectives on corporate OSS approaches.

COMMERCIAL FREE AND OPEN SOURCE SOFTWARE: KNOWLEDGE PRODUCTION, HYBRID APPROPRIABILITY, AND PATENTS
GREG R. VETTER, 77 *Fordham Law Review* 2087, 2088–94 (2009)

Compare Robert Jacobsen [the OSS developer who served as plaintiff in *Jacobsen v. Katzer*] to MetaCarta, a company involved in both proprietary software development and related services for its users, and involved with certain niche open source communities relating to software for displaying geographic information. I choose MetaCarta as a stylized example

[28] Klint Finley, *Why 2018 Was a Breakout Year for Open Source Deals*, Wired, December 23, 2018.

because it is not involved in any litigation of which I am aware. But it has a noteworthy approach to its role in the greater world of free and open source software (FOSS) development. MetaCarta contributes some of its software to the FOSS community by acting as the organizing hub for three FOSS projects. This is not unheard-of. More uniquely, however, it also actively seeks a small portfolio of patents in related areas of software technology. Following a trend, MetaCarta is backed by venture capital investors while explicitly embracing the FOSS movement. Compared to a for-profit entity such as MetaCarta, Robert Jacobsen is a sympathetic figure for a court. He was a volunteer developing FOSS with public benefit spillovers. His motivations likely fit within some of the typically offered explanations for FOSS volunteerism: to scratch a technological itch; to have fun; to participate in a community; to learn; or to enhance career prospects. MetaCarta's motivations are those of a for-profit firm with investors hoping for return and market share. While software patenting has become common among information technology companies, much of the FOSS movement would see it as nonbeneficial. MetaCarta, however, represents a trend: "commercial FOSS" that hybridizes proprietary software appropriation techniques with conventional FOSS volunteerism-centric development.

Jacobsen and MetaCarta illustrate a dualism in FOSS that channels the knowledge production and distribution influences of the movement and could impact the perspective of future courts as they engage other licensing law issues likely to arise … On one side of the dualism is the free software strand within the FOSS movement, while on the other is the open source strand. Each correlates to different licensing models and to different practices to gather satisfaction from writing and supplying software. The free software strand would typically use licenses with a mechanism known as copyleft to ensure that the original license conditions (often requiring source code availability and sometimes prohibiting ongoing royalties) remain in place for downstream versions of the software. With this, appropriating value from the software is biased toward services and other economic complements whenever the FOSS developer needs value to accrue to her in a pecuniary fashion.

Jacobsen's story is the narrative of the stylized FOSS developer who codes and shares for nonpecuniary satisfactions. Jacobsen's group did not use a copyleft license, but many similarly situated groups do so. For historical reasons developers often choose the Free Software Foundation's (FSF) General Public License (GPL), which is a strong copyleft license locking the software under its scope into a development mode characterized by source code availability and a prohibition against ongoing royalties to run the software.

MetaCarta's narrative is that of open source software development within a for-profit company. It applies an attribution-only license to the projects it stewards, meaning that others can deploy or use the software however they wish so long as such later deployment gives attribution to the software's originators. Open source developers sometimes start projects under an attribution-only license to allow for the future involvement of a company under a proprietary or hybridized model. The attribution-only license allows for the possibility to later release the software under either the GPL or as proprietary software, or perhaps as both in a dual-licensing strategy. MetaCarta has the twist of involving itself with patents. Other commercial FOSS entities, however, use kindred mechanisms, such as dual licensing, to rig an appropriability mix that allows some benefits of FOSS development to contribute to the prospects of the entity. Hopefully, as a result, the entity is therefore also a better (more financially viable) steward for the FOSS projects.

FOSS's influences on knowledge production and distribution … must be considered in light of the free software/open source dualism, but also in light of appropriability. The weight of the literature to date treats FOSS as a nonmarket, peer-production method of developing and distributing new knowledge. FOSS has generated new knowledge in the sense of new collaboration models for software development and market deployment; inspired other movements, such as Creative Commons or free culture generally; and it provides or supports numerous technology platforms, including important elements of the Internet's past and future development.

This impressive scorecard of knowledge production is bronzed by FOSS benefits in knowledge distribution. Simply put, FOSS created a sea change in the availability of source code to study and learn coding and software technology at every level of complexity and in an incredibly diverse array of languages and information technology environments. In other ways, however, the benefits of FOSS are less clear. Superior code quality, in terms of lower defects and greater resistance to problems, is often argued to be a FOSS benefit for structural reasons. Empirical evidence on the point, however, is mixed, although many high-profile FOSS projects are clearly of very high quality. A reframed question is more to the point: is the quality of software developed with the methods of the FOSS movement of higher quality compared to traditional proprietary software development? If so, this is a part of FOSS's contribution to knowledge creation for the information technology ecosystem.

Software is of greater benefit not only if its quality is high, but also if it provides superior functionality. Often superior functionality means new functionality; that is, technology innovation from some programmatic processing, presentation, or interfacing that is novel and heretofore not in existence within information technology. The creation of new nonplatform software functionality may not yet be a primary strength of FOSS development. Assuming this is true, it raises a knowledge production question for FOSS: can the movement gain momentum in generating new nonplatform functionality as opposed to primarily moving functionality from one platform to another, or commoditizing existing software products?

Is the mechanism to gain this momentum in the nonpecuniary satisfaction of volunteer developers coupled with the leveraging of economic complements under the free software approach? Or, is the path in open source appropriability with commercial FOSS experiments such as MetaCarta?

These questions are not in a vacuum because other new appropriability mechanisms for software have mainstreamed in the last decade. Thus, the traditional models, such as the proprietary software product vendor model and the custom software developer model, now compete with advertising-supported software and web-delivered software as a service.

A History of IBM's Open-Source Involvement and Strategy

Peter G. Capek, et al., 44 *IBM Systems Journal* 249 (2005)[29]

The origins and principles of free software and of open-source software (OSS) may lead the casual observer to conclude that they are a world apart from—if not opposed to—more traditional software development, use, and evolution. An alternative view sees OSS

[29] Reprint Courtesy of International Business Machines Corporation, © 2005 International Business Machines Corporation.

as essentially an alternative business model which provides types of flexibility, opportunity, and benefits different than those provided by the conventional model. IBM was among the earliest of the major computer companies to embrace opensource software and was probably the first to realize that doing so could be consistent with our business goals. Indeed, a problem with which IBM has long contended is that of how to provide to our customers internally developed software that was not planned to be a product, without the inevitable support and product issues.

In December of 1998, an effort was first made to understand the broad strategic implications for IBM of open-source software. At that point, it was clear that the OSS phenomenon was taking hold in a substantial way. Most visibly, Linux was starting to appear widely in the media, but more importantly, parts of our customer organizations were starting to pay attention, with Linux reportedly being used in some cases without the involvement or blessing of corporate IT organizations. Quickly, we realized that whether this evolved into an important force or whether it remained a minor fad, the potential was such that it was important to understand its implications for our customers and for us and be able to respond appropriately. Before 1999, our involvement was on a case-by-case basis.

...

An important issue was the quality of software that was produced by open-source communities and their collaboration. Much of IBM's product software development was historically quite structured, with substantial initial planning and design, followed by implementation, unit and system testing phases, and of course ongoing support and maintenance. Many at IBM had the impression—partly from what appeared in the business and technical press—that open-source software efforts were closer to the other end of the spectrum in terms of structure and management discipline, and they were accordingly skeptical that the quality of the open-source software produced could be sufficient to be relevant to us and our customers.

These early fears turned out to be unfounded. Even at that time (ca. 1999), the quality of the software from the open-source projects investigated was impressive. It was clear that this development style attracted very skilled developers, and that the overlap between developers and users of a particular OSS project made possible excellent and open communication, rapid development cycles, and intensive real-environment testing, ultimately producing software that was often very good and sometimes excellent by our standards. At the same time, it was immediately clear that there were important areas where IBM's large and excellent technical community could make significant contributions, having substantial experience, and in doing so, our customers could be helped to reap the benefits of our expertise in an open context. In more recent years, the possibility of inverting the model has been investigated, whereby our proprietary development activities can benefit from what has been learned from the open community.

...

From the outset, it was clear that a host of legal and business considerations needed to be understood if IBM was going to participate in any OSS activities in a meaningful way. Much of the participation and development of OSS at that time was done by individuals acting on their own. There were some early efforts that were more organized and which involved small companies, but these were, for the most part, companies organized around their opensource participation. A few notable examples included companies that were using open source in their own operations and contributing enhancements and development to it for the broader good.

IBM, of course, had a large software business, which could not be put at risk; therefore, it was important that any risks associated with OSS be identified, and the legal, strategic, and business issues surrounding open source and its licensing be understood. Where needed, procedures would have to be established to ensure that our participation was principled and appropriate.

…

More generally, a strategy was planned that allowed us to add value for our customers in the areas where our ability to do so was greatest. This was clearly in the broad area of what is called middleware, and not in operating systems, because our enterprise customers benefit more directly from middleware functions than from operating-system functions; analogous statements can be made in other areas. Consequently, our strategy for open-source participation was one which effectively minimized the distinctions at the operating-system level and allowed us to retain the ability to differentiate where we could have the greatest impact.

…

Complexity sets in with software because most substantial open-source software has many authors and is developed in a collaborative and informal manner by people with no particular legal relationship. For these projects, it is often difficult years later to know reliably whether the person granting a license had the right to do so. For instance, was a particular contributor the author of the code, and did he have the right to grant a license, or did his employer acquire that right when he wrote it? Although some projects, including those under the Free Software Foundation, have long required assignment of copyright by each contributor including written signatures, this has not been a universal practice. Recently, more software community leaders have recognized the importance of creating clarity of code “pedigree” and rights, and IBM has worked to assist some open-source projects to increase the rigor of their processes in this area. Examples of these efforts are the Linux kernel and its Developer’s Certificate of Origin and the Apache Software Foundation and its Contributor License Agreement.

Another legal consideration was the proliferation of licenses used for open-source projects. None of these licenses had been interpreted by any court, and they varied greatly in terms of their legal robustness and completeness. Many of them were unclear with respect to the granting of intellectual property rights. As a commercial organization, we felt it was important to encourage a model in which commercial products could be based on open-source efforts, and we needed to identify a license that would permit such a model. Thus, IBM created, used, and encouraged the use of, what is now known as the CPL, or Common Public License. This license has been well received by the community, and its use seems to be increasing. It has been certified as an open-source license by the Open Source Initiative. Our goals in creating this license were to provide a means for commercial organizations to base products on open-source efforts, to encourage a common OSS practice of making modifications and enhancements available as source code, and to provide a model which could help to shape other open-source licenses. In our opinion, this license provides a good balance between open-source and commercial efforts and encourages enhancements to open-source projects.

Notes and Questions

1. *OSS dualism*. Explain the "dualism" that Vetter observes in the OSS world.
2. *Commercial hurdles*. What strategic and business assumptions did IBM have to overcome before it was convinced that adoption of OSS was a sensible commercial move?
3. *Small vs. large*. Compare the OSS strategies of Robert Jacobsen, MetaCarta and IBM. How do they differ? What are their similarities?
4. *Software pedigree*. Why does IBM raise the issue of a software program's "pedigree" as a concern? The authors mention that the FSF once required that contributors assign copyright in their software contributions to the FSF. Why would they do that? IBM and others elected not to require such assignments, but developed alternative methods of ensuring software pedigree. What do you think these alternative methods entailed?
5. *License proliferation*. One of the challenges that IBM notes is the proliferation of OSS licenses – a problem that OSI was considering at about the same time (see Section 19.2.2). In IBM's case, this concern resulted in IBM developing its own form of OSS license. Why did IBM take this approach? Some observers have called company-specific OSS licenses "vanity licenses." What benefits can you see in allowing every company to create its own form of OSS license versus using a small set of widely adopted OSS licenses?

19.3.2 *Integrating OSS with Commercial Products*

How, precisely, should OSS be integrated with commercial products? This section addresses some of the practical legal and contracting issues that arise when integrating OSS and commercial software, both for internal use within an enterprise and in a software product or service for distribution to others.

19.3.2.1 Considerations for Using OSS in a Corporate Enterprise

Corporate IT managers who are considering the use of OSS products within the enterprise must consider a host of technical issues including the following:

1. Do the enterprise's internal IT staff have the expertise to install and operate the OSS software without external assistance, or must external consultants be hired?
2. How will the OSS be integrated with existing systems?
3. Does the OSS meet all data security and privacy requirements imposed by internal corporate policies as well as external regulatory and licensing agencies (e.g., HIPAA for medical records)?
4. Is it necessary to customize the OSS for internal usage, or will it satisfy internal needs in its current form?
5. Is a commercial substitute available at a reasonable cost?
6. How important is the availability of technical support, help, maintenance and updates? Can these be provided by internal IT staff?
7. What experiences have other similarly situated enterprises had with this OSS product?
8. What licensing restrictions surround the use of the OSS?
9. Is there any chance that the OSS, or a system that includes the OSS, will be shared with third-party partners, collaborators or affiliates in a manner that will constitute "distribution" of the code triggering OSS licensing requirements such as source code availability?
10. How closely is the OSS code integrated into proprietary code? Can it introduce security vulnerabilities?

19.3.2.2 Considerations for Incorporating OSS into a Distributed Product

A host of OSS modules, libraries and applications that perform a wide range of functions are available for minimal or no cost. It is tempting to use these OSS programs in commercial products, as they reduce costs and accelerate development schedules. What's more, many software engineers are familiar with OSS code that they used (or wrote) in graduate school or at prior jobs. Product developers and managers, however, should consider a variety of factors before permitting OSS to be incorporated into a commercial hardware or software product.

1. What type of license is the OSS covered by? A "viral" license such as GPL, or a license that requires broad patent grants, such as GPL or Mozilla, may be disqualifying. Permissive licenses such as BSD, MIT and Apache may be more acceptable. Careful study of the projected integration of the OSS code into proprietary software should be made before accepting OSS licensed under the LGPL.
2. Is the larger product intended to be released on an OSS basis? If so, then a "viral" license such as GPL may not be as problematic as it might be if the larger product were intended to be released on a proprietary basis.
3. How important is the support of an OSS community of developers to the acceptance, adoption and dissemination of the product?
4. How will the product be supported and updated? Does the internal staff have the ability to support the OSS code?
5. Bearing in mind that most OSS comes with no warranty or liability, what risks are involved in the operation of the product, and what harm might arise if the OSS malfunctions? Is the product a pacemaker, a nuclear reactor controller or a new Solitaire app?
6. Can the OSS be validated in terms of security, privacy and regulatory compliance?
7. Is there a reasonably priced commercial alternative to the OSS code?
8. How closely is the OSS code integrated into proprietary code? Can it introduce security vulnerabilities?

19.3.2.3 Required Notices and Licensing Terms

Even companies like Apple that have traditionally favored the use of proprietary code have integrated OSS with some of their commercial software products. When doing so, a company must be careful to disclose any applicable OSS licensing terms in its relevant product licensing agreements, just as it must for any other third-party software integrated into its products (see Section 9.2.1.2). Below is an example of the text that Apple includes in one of its recent software license agreements to address OSS requirements.

APPLE BIG SUR MACOS LICENSE (2020)[30]

Open Source. Certain components of the Apple Software, and third party open source programs included with the Apple Software, have been or may be made available by Apple on its Open Source web site (https://www.opensource.apple.com/) (collectively

[30] www.apple.com/legal/sla/docs/macOSBigSur.pdf.

the “Open-Sourced Components”). You may modify or replace only these Open-Sourced Components; provided that: (i) the resultant modified Apple Software is used, in place of the unmodified Apple Software, on Apple-branded computers you own or control, as long as each such Apple computer has a properly licensed copy of the Apple Software on it; and (ii) you otherwise comply with the terms of this License and any applicable licensing terms governing use of the Open-Sourced Components. Apple is not obligated to provide any updates, maintenance, warranty, technical or other support, or services for the resultant modified Apple Software. You expressly acknowledge that if failure or damage to Apple hardware results from modification of the Open-Sourced Components of the Apple Software, such failure or damage is excluded from the terms of the Apple hardware warranty.

Certain software libraries and other third party software included with the Apple Software are free software and licensed under the terms of the GNU General Public License (GPL) or the GNU Library/Lesser General Public License (LGPL), as the case may be. You may obtain a complete machine-readable copy of the source code for such free software under the terms of the GPL or LGPL, as the case may be, without charge except for the cost of media, shipping, and handling, upon written request to Apple at opensource@apple.com. The GPL/LGPL software is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY, without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. A copy of the GPL and LGPL is included with the Apple Software.

Notes and Questions

1. *Enterprise versus product*. How do the considerations for IT managers considering using an OSS program within an enterprise differ from the considerations for software product developers considering using an OSS program in a product for distribution? What should be the greatest concerns for each?
2. *Dual licensing*. It is important to remember that neither the GPL nor any other OSS license requires the owner of a copyright in a software program to assign or give up that copyright. Accordingly, the owner of a software program that releases it under an OSS license retains copyright in that program. And, as such, the owner may decide to release the program under both an OSS license *and* a proprietary license. Why would a copyright owner do this?
   Companies like MySQL have developed “dual-licensing” programs. They make their software available for free on an OSS basis (sometimes under the GPL), but also offer a commercial licensing option that comes with user support, maintenance and a warranty. This option is often attractive to corporate IT managers. While they would save some money by using the free OSS version, they also value the ability to get support from the software vendor. What drawbacks might a software vendor face with a dual-licensing approach?

Problem 19.2

You are the general counsel of FishFry Corp. (NYSE: FFC), a publicly traded Seattle-based manufacturer of deep-frying equipment for the fast-food restaurant market. FFC’s flagship product is the FF-1000 (so-named because it heats the cooking oil to a temperature of 1000°F). The

FF-1000 uses a sophisticated proprietary sensor-plus-software system to monitor and adjust cooking temperature during use. Unfortunately, due to the "health food craze" that is sweeping the nation, the deep-fried food market is suffering and FFC's customers are not inclined to upgrade their equipment. Worse, FFC's biggest competitor, HeißFrei GmbH, a German manufacturer, has just released the SuperHeiß-1001, which cooks at one degree hotter *and* is priced $100 less than the FF-1000. But there may be hope! FFC's chief engineer, Haddock Sturgeon, just came by your office and mentioned that a well-known thermodynamics engineer at the University of East Nevada recently released a new, highly efficient, open source code temperature control algorithm on his website. The software was developed as part of a research project on geothermal energy, but Haddock is pretty sure that his team can make any necessary modifications and integrate it into the FF-1000 control system. Best of all, it's free, and it will make the FF-1000 15 percent more energy efficient, a big selling point for customers. What questions and concerns do you have regarding Haddock's plan?

19.3.3 *OSS Due Diligence*

The issue of open source "contamination" of proprietary code often arises in the context of acquisition transactions. That is, when an acquirer is considering the purchase of a target company or a division of another company, it may wish to understand the licensing regimes governing the target's products. This is particularly important if the target is a small company or university spinoff, in which software developers and engineers are accustomed to working with OSS code.

The bulk of an acquirer's "due diligence" in considering such an acquisition should be technical and include code reviews and walkthroughs with the target's technical personnel. But legal due diligence is also advisable. This includes reviewing the licensing agreements that apply to the target company's products.

EXAMPLE: OPEN SOURCE REPRESENTATION AND WARRANTY

"Open Source Materials" means all software or other material that is distributed as "free software," "open source software" or under a similar licensing or distribution model, including, but not limited to, the GNU General Public License (GPL), GNU Lesser General Public License (LGPL), Mozilla Public License (MPL), BSD Licenses, and the Apache License.

Open Source Code. Section __ of the Disclosure Schedule lists all Open Source Materials that Company has utilized in any way in the Exploitation of Company Offerings or Internal Systems and describes the manner in which such Open Source Materials have been utilized, including, without limitation, whether and how the Open Source Materials have been modified and/or distributed by Company. Except as specifically disclosed in Section __ of the Disclosure Schedule, Company has not (i) incorporated Open Source Materials into, or combined Open Source Materials with, the Customer Offerings; (ii) distributed Open Source Materials in conjunction with any other software developed or distributed by Company; or (iii) used Open Source Materials that create, or purport to create, obligations for Company with respect to the Customer Offerings or grant, or purport to grant, to any third party, any rights or immunities under Intellectual Property rights (including, but not limited to, using any Open Source Materials that require, as a condition of Exploitation of such Open Source Materials, that other Software incorporated into, derived from or distributed with such Open Source Materials be (a) disclosed or distributed in source code form, (b) licensed for the purpose of making derivative works, or (c) redistributable at no charge or minimal charge).

Notes and Questions

1. *The importance of OSS review*. Why is it important for an acquirer to understand the degree to which a target company employs OSS in its products?
2. *Black Duck*. In many cases, the recollections and records of a target company's personnel are inadequate to identify the OSS code within a large product code base. Since the early 2000s, products have been available to scan a code base to detect OSS code included within it, and to identify the applicable licensing terms. One early entrant into this market was Black Duck Software, a firm formed in 2002 by former Microsoft employees and acquired by Synopsis in 2017. Black Duck deploys algorithms to scan a code base for incidences of more than 2,700 known OSS programs.
   OSS proponents have charged that firms like Black Duck exist only to spread fear, uncertainty and doubt (FUD) about OSS. What do you think? Is OSS scanning/auditing a useful service, or merely a ploy by proprietary software giants to discredit OSS?

19.4 PATENT PLEDGES

The previous sections of this chapter have focused largely on public licenses of copyrighted material – online content and software. While several OSS licenses include explicit or implicit terms relating to patents, these are not their primary focus. Yet the rise of commercial OSS in the 1990s, particularly the Linux operating system, motivated several large companies to eliminate the potential barriers to large-scale adoption of OSS software presented by their patent portfolios. The solution that they arrived at were public-facing patent "pledges."

PATENT PLEDGES: BETWEEN THE PUBLIC DOMAIN AND MARKET EXCLUSIVITY
Jorge L. Contreras, 2015 *Michigan State Law Review* 787

Patent pledges are "[public] commitments voluntarily made by patent holders to limit the enforcement or other exploitation of their patents." These pledges encompass a wide range of technologies and firms: from promises by multinational corporations like IBM and Google not to assert patents against open source software users; to commitments by developers of industry standards to grant licenses on terms that are fair, reasonable, and non-discriminatory (FRAND); to the recent announcement by Tesla Motors that it will not enforce its substantial patent portfolio against any company making electric vehicles in "good faith."

Despite this diversity in content and form, patent pledges share a number of unifying features. The public nature of patent pledges distinguishes them from the broad array of formal licenses that patent holders routinely grant in commercial transactions. First, patent pledges are not made to direct contractual counterparties or business partners, *but to the* *public at large*, or at least to large segments of certain markets. Second are the motivations that lead patent holders to make patent pledges. In general, these motivations fall into two broad categories: (1) inducing other market participants to adopt, and make investments in, a standardized technology or other common technology platform; and (2) "soft" factors including communitarianism, altruism, and the desire for improved public relations. Broadly speaking, this Article addresses the first category of pledges, those that are made with an intention to induce movement in the relevant technology market, and which I have termed "actionable" pledges.

To understand the reasons that patent holders make patent pledges, it is first important to consider the beneficial market-wide effects that patent pledges can have. For example, technical interoperability standards enable devices manufactured by different vendors to interoperate automatically and without significant user intervention. The Wi-Fi wireless networking suite of standards is a good example. Any computer, tablet, smart phone, or other device that implements the relevant Wi-Fi standard can communicate with any other device that implements the same standard. The manufacturers of those devices need not interact at all during the development and manufacturing of their respective products. So long as two devices comply with the relevant standard, they can communicate with each other.

The benefits that can be achieved through widespread product interoperability are known as "network effects" and generally increase as the number of compatible devices grows. The interoperability of different vendors' products opens markets for new products and services, fostering innovation, competition, consumer choice, and economic growth. As observed by the principal U.S. antitrust agencies, standards enabling product interoperability "are widely acknowledged to be one of the engines of the modern economy." The same holds true for some software platforms, particularly those that are characterized by open application program interfaces (APIs) or are distributed in open-source form. The broad availability of such software platforms can give rise to market-wide cost savings and efficiencies, and can promote consumer choice and competition, as exemplified by the Linux and Android operating systems.

Patent pledges create an environment in which multiple firms are more likely to adopt particular standards or open-technology platforms, resulting in greater product interoperability and increased network effects. Why? This is because the holder of patents, which might otherwise be used to block a competitor from developing and selling a compatible product, commits to limit the use of those patents. This commitment might come close to contributing the patent to the public domain, for example, by pledging not to enforce a software patent against any company with fewer than twenty-five employees. At the other end of the spectrum, the pledge might simply be to grant royalty-bearing patent licenses on terms that are "fair, reasonable and non-discriminatory." In both cases, patent owners limit their statutory right to enforce their patents. By doing so, they seek to induce market participants to adopt *their preferred* standards or technology platforms. In other words, such pledges create a "safe space" in which product development and innovation can flourish with a reduced threat of patent enforcement. Such commitments thus benefit the market broadly, but also guide the market toward the patent holder's own products and technologies, which benefits the patent holder. Patent pledges thus have the potential to produce a number of beneficial market effects, which alone should be sufficient reason to respect and enforce them.

However, there is another reason that patent pledges, as a general rule, should be treated as legally enforceable obligations. This justification is based on the reliance of other market actors on these pledges. Manufacturers who rely on a patent holder's promise not to block the sale of a product will often make costly investments on that basis. These investments could include product design and development, marketing, materials, capital equipment, information technology, employee training, and supply chain management. Once such investments have been made, the manufacturer is said to be "locked-in" and cannot switch to an alternative technology without significant, and potentially prohibitive, cost. Thus, it is important to enforce the patent holder's pledge to protect other market actors who have relied on those pledges in making investments that, in the end, are likely to have a socially beneficial effect.

Various theories have been advanced regarding the most appropriate legal framework for enforcing patent pledges. These include common law contract, antitrust law, patent misuse, and other theories based in equity and property law. Each of these approaches has theoretical or practical drawbacks that I have previously discussed at length. As an alternative, I have proposed a new theory termed "market reliance," which begins with the equitable doctrine of promissory estoppel and adds to it a rebuttable presumption of reliance adapted from the "fraud-on-the-market" theory under Federal securities law. The market-reliance approach, which focuses on a patent holder's behavior-inducing promise to the market, may enable patent pledges to be recognized and enforced without the need to prove the elements of contract formation, antitrust injury or specific reliance.

But as I have also explained elsewhere, any reliance-based approach requires that the relevant promise have some degree of visibility to the market, even if individual market actors are not aware of specific pledges made with respect to specific patents. Thus, pledges that are posted on a web site and taken down the next day, or are substantially changed after they are made, raise questions regarding their later enforcement. If an initial announcement attracted sufficient public attention, such pledges might influence markets significantly. Yet if their appearance and disappearance went unnoticed, then it is likely they would have no impact on the market. And, of course, most situations will fall somewhere between these two extremes.

Patent pledges have already shaped critical technology markets and enabled the interoperability of a vast range of products and services. However, as patent litigation in these markets has increased, the premises and assumptions underlying these pledges have begun to show stress. I have proposed both a theoretical framework (market reliance) and a practical resource (the pledge registry) that, it is hoped, will solidify the legal foundation for this critical middle ground between the public domain and market exclusivity.

ALL OUR PATENT ARE BELONG TO YOU!

Elon Musk, CEO [Tesla Motors], June 12, 2014

Yesterday, there was a wall of Tesla patents in the lobby of our Palo Alto headquarters. That is no longer the case. They have been removed, in the spirit of the open source movement, for the advancement of electric vehicle technology.

Tesla Motors was created to accelerate the advent of sustainable transport. If we clear a path to the creation of compelling electric vehicles, but then lay intellectual property landmines behind us to inhibit others, we are acting in a manner contrary to that goal. **Tesla will not initiate patent lawsuits against anyone who, in good faith, wants to use our technology**.

When I started out with my first company, Zip2, I thought patents were a good thing and worked hard to obtain them. And maybe they were good long ago, but too often these days they serve merely to stifle progress, entrench the positions of giant corporations and enrich those in the legal profession, rather than the actual inventors. After Zip2, when I realized that receiving a patent really just meant that you bought a lottery ticket to a lawsuit, I avoided them whenever possible.

At Tesla, however, we felt compelled to create patents out of concern that the big car companies would copy our technology and then use their massive manufacturing, sales and marketing power to overwhelm Tesla. We couldn't have been more wrong. The unfortunate reality is the opposite: electric car programs (or programs for any vehicle that doesn't burn hydrocarbons) at the major manufacturers are small to non-existent, constituting an average of far less than 1% of their total vehicle sales.

At best, the large automakers are producing electric cars with limited range in limited volume. Some produce no zero emission cars at all.

FIGURE 19.10 Elon Musk, the flamboyant CEO of Tesla Motors, pledged all of the company's patents in a 2014 blog post.

Given that annual new vehicle production is approaching 100 million per year and the global fleet is approximately 2 billion cars, it is impossible for Tesla to build electric cars fast enough to address the carbon crisis. By the same token, it means the market is enormous. Our true competition is not the small trickle of non-Tesla electric cars being produced, but rather the enormous flood of gasoline cars pouring out of the world's factories every day.

We believe that Tesla, other companies making electric cars, and the world would all benefit from a common, rapidly-evolving technology platform.

Technology leadership is not defined by patents, which history has repeatedly shown to be small protection indeed against a determined competitor, but rather by the ability of a company to attract and motivate the world's most talented engineers. We believe that applying the open source philosophy to our patents will strengthen rather than diminish Tesla's position in this regard.

Notes and Questions

1. *Pledge plus public license*. Like the Creative Commons licensing tags, some patent pledges include both a short public pledge statement as well as a public license containing more detailed terms. This approach was used, for example, by the Open COVID Pledge (www.opencovidpledge.org),[31] under which a number of IP holders pledged patents and copyrights to fight the COVID-19 pandemic. What are the advantages of this two-tiered pledge approach? Can you think of any disadvantages?
2. *Tesla's pledge*. Do you think that the pledge made by Elon Musk in a blog post legally binds his company, Tesla Motors? Why do you think that Musk approached this important grant of rights in this relatively informal manner? As it turns out, Tesla's legal department also had concerns with Musk's pledge, and a year later reissued it on Tesla's corporate website in more robust legal terms. Was this revision necessary?
3. *Motivations for pledges*. What do you think motivated Tesla to make its pledge? Why did it sacrifice potential royalty income, or market exclusivity, for no apparent financial gain? Likewise, why did several large IP holders like IBM, Microsoft, Amazon and Facebook make the Open COVID Pledge? Do you think their motivations differed from Tesla's motivation to pledge its electric vehicle patents?[32]

[31] See also Jorge L. Contreras, *The Open COVID Pledge: Design, Implementation and Preliminary Assessment of an Intellectual Property Commons*, 2021 Utah L. Rev. 833 (2021).

[32] Jorge L. Contreras, *Patent Pledges*, 47 Ariz. St. L.J. 543, 573–92 (2015).