7

The Mozilla Public License (MPL)

The Mozilla Story

In the late 1990s, Netscape was facing a serious problem. Its browser, the Netscape Communicator, was rapidly losing market share to Microsoft's Internet Explorer, and it was difficult for them to define a competitive business case to justify ongoing development and licensing of their proprietary browser. Rather than simply shut down development, however, Netscape decided to turn it into an open source project and to license their software to the public under an open source license. But which license?

Netscape resisted using an academic license because, in the company's opinion, such licenses don't go far enough to return certain modified code back to the community. (This history is discussed in much more detail at *www.mozilla.org*.) They realized that academic licenses allow "middlemen" to remove improvements from the free software commons—and they didn't want that to happen.

They also resisted using the GPL for four important business and legal reasons. First, they believed that the GPL was incompatible with certain obligations they had under other licenses for software they had previously incorporated into their browser. Second, they weren't sure if the GPL was consis-

tent with cryptographic code regulated by U.S. law. Third, they weren't sure what their reciprocity obligations would be under the GPL for other Netscape products (particularly servers), and they wanted to be sure that other software of theirs could remain proprietary. And fourth, they were concerned that other companies would decline their software if the GPL were used.

Netscape even considered using the LGPL because that license seemed to narrow the risk that software that merely interacted with their browser would come under the reciprocity provision. But it too was rejected.

A Netscape executive and attorney, Mitchell Baker, who understood both the structure of Netscape software and the legal requirements, wrote a new open source license to address these problems. The resulting Mozilla Public License (the MPL), has been the model for most of the important commercial open source licenses that followed. Next to the BSD and the GPL, the MPL is the most influential open source license. Baker went on to become the *Chief Lizard Wrangler* of the Mozilla open source project.

The MPL is a serious license. I will direct much less criticism to the structure and terms of the MPL in this book than to the other licenses I've already written about, because the MPL is a high-quality, professional legal accomplishment in a commercial setting.

One of the challenges to writing about licensing in a book not specifically written for licensing professionals is to make a very dull subject interesting. For those readers who are skilled computer programmers, compare my challenge to that of an engineer who wants to explain C++ programming or the TCP/IP stack to the public.

How do I explain an open source license like the MPL deeply enough to make my description accurate without quot-

ing pages of legal provisions and explaining how courts will interpret them? The most recent version (1.1) of the MPL is copied in the Appendices. Obviously I don't have to reprint each section seriatim and translate it into colloquial English, nor parse each sentence so that you can recognize a derivative work or a collective work when you see it.

So instead what I will do is paint a broad picture. I'll describe how the MPL and similar licenses from many other commercial companies are structured. I'll highlight things you and your attorney should look for in such licenses when accepting software under them, and what you need to consider when you modify those licenses for use in distributing your own software.

The MPL Reciprocity Bargain

Although the MPL is a much longer license than the others I've discussed, when you get beyond the complex words, its reciprocity provisions can be paraphrased very simply as follows: *If you create and distribute a Modification to one of the files containing Original Code or previous Modifications, or create and distribute a new file containing Original Code or previous Modifications, those files must be released as Modifications under the same MPL license.*

Your newly released files become *Modifications* for future licensees. One can recognize in this recursive definition how a chain of title is created to ever-more-modified derivative works, with each Contributor adding to the chain. But here, unlike with previous licenses, the MPL deals with *files containing derivative works* rather than *derivative works* more broadly.

This calls for precise definitions, which the MPL provides. Here are four of them:

> *"Contributor" means each entity that creates or contributes to the creation of Modifications. (MPL section 1.1.)*
>
> *"Covered Code" means the Original Code or Modifications or the combination of the Original Code and Modifications, in each case including portions thereof. (MPL section 1.3.)*
>
> *"Modifications" means any addition to or deletion from the substance or structure of either the Original Code or any previous Modifications. When Covered Code is released as a series of files, a Modification is:*
> *A. Any addition to or deletion from the contents of a file containing Original Code or previous Modifications.*
> *B. Any new file that contains any part of the Original Code or previous Modifications. (MPL section 1.9.)*
>
> *"Original Code" means Source Code of computer software code which is described in the Source Code notice required by Exhibit A as Original Code, and which, at the time of its release under this License is not already Covered Code governed by this License. (MPL section 1.10.)*

Such definitions are extremely important in software licenses. You have already seen how words in simple academic licenses, and even some words in the venerable GPL, are confusing and subject to misinterpretation. Words without definition are ambiguous; there is no reliable way to predict how the parties—or a court—might interpret a license without clear definitions when performance under it is called for or questioned.

One way to deal with that problem is to rely on *terms of art*, words that will be understood by the parties and by courts because they are defined in the legal lexicon or by statute. That is why I have been so adamant in this book about using the terms *collective works* and *derivative works* precisely. Those terms are defined by statute for all lawyers to understand

(although few really do), and if we use them consistently we'll at least all mean the same thing. We can also use court decisions from similar cases to help us predict how the courts will interpret certain terms of art in our own licenses.

Commercial open source licenses like the MPL, in addition to using legal terms of art precisely, often rely on their own definitions of terms. (The GPL did that for the term *Program*.) Those *definitions* must be read carefully because, in license interpretation and enforcement, they often take precedence over the *terms of art*. For example, the four definitions I quoted from the MPL above distinguish carefully between *Covered Code* and *Original Code*; the latter is included in the former. Note that the term *Modifications* is defined in light of *Original Code* in its first sentence and *Covered Code* in its second sentence. We must parse very carefully to know our reciprocity obligations under such licenses.

Contributors and Modifications

I described in the first chapters of this book how open source development is a continuous process. Contributors and distributors enhance and improve software at each step by creating collective and derivative works. That explanation was necessary because the BSD and earlier licenses weren't explicit about it. Collaborative open source development progressed under those licenses without the licenses mentioning the process at all.

The MPL defines this process much more precisely in sections 2 and 3. Open source development starts with *Original Code* supplied by an *Initial Developer* (in the first instance Netscape Corporation, although the MPL is a *template license*) who licenses all relevant open source rights to *You*. *You* make *Modifications* and become a *Contributor*. As a *Contributor*, *You*

are required by the reciprocity terms of the MPL to license all relevant open source rights for *Your Modifications* to everyone under the MPL, and to provide *Source Code*. *You* also received, by that recursive MPL license, a license to all the *Modifications* made by earlier *Contributors*. (Notice the MPL-defined terms are the italicized nouns with capital letters throughout this paragraph.)

Once again, because of the MPL's definitions, this continuous enhancement process applies not to derivative works as a whole but separately to each *file* containing *Modifications*. (The word *file* is not defined in the MPL. This word is a *term of art* from the computer field; we must rely on experts to tell a court what that word means; I'm confident, though, that every reader of this book has a clear idea what the word *file* means and would recognize a *modified file* even without an MPL definition for it.)

The license grant under the MPL was structured so as to apply more narrowly than the GPL and other previous licenses. It licenses *files* to be modified, not *programs* to be turned into *derivative works*. This has one major consequence for those who create *Larger Works*, works that combine Covered Code with code not governed by the terms of the MPL License. Under copyright law, such *Larger Works* might be *derivative works*, depending upon the nature of the *combination* of software being created; but under the MPL's definitions, a *Larger Work* has more limited implications:

> *You may create a Larger Work by combining Covered Code with other code not governed by the terms of this License and distribute the Larger Work as a single product. In such a case, You must make sure the requirements of this License are fulfilled for the Covered Code. (MPL section 3.7.)*

You are only obligated to apply the MPL's license restrictions to *files* containing *Original Code* or *Modifications*. The rest of the files—your own files—are not affected by the reciprocity obligation, even if you have created a derivative work by adding your own files.

Any licensee intending to create *Larger Works* would be wise to consult an attorney to decide whether such a work is considered a *Modification* that must be contributed back under the MPL, or whether there are any other license obligations still to honor for the *Covered Code*.

In a way, the MPL is a kind of compromise between the academic and reciprocal license models. Reciprocity under the MPL is defined narrowly so as to encourage the use of open source software as building blocks to create *Larger Works*. (Those *Larger Works* may even be derivative works under copyright law; that doesn't matter.) Those *Larger Works* may be open or proprietary; with respect to them, the MPL acts like an academic license. But the individual building blocks are licensed with reciprocity obligations. If you distribute improvements to those building blocks, you must license those improvements under the MPL as open source software.

The MPL and Patents

The MPL also deals with patents much more thoroughly than any preceding open source license.

To review, a patent is a grant under power of law of the right to exclude others from making, using, selling, offering to sell, or importing certain specifically claimed inventions. The claims in a patent can be licensed to others. The MPL actually defines *Patent Claims* more precisely as "including, without limitation, method, process, and apparatus claims in any patent Licensable by grantor." (MPL section 1.10.1.) This is

consistent with the types of utility patents actually granted by the U.S. Patent and Trademark Office. Fortunately, these technical distinctions among types of claims won't be important for us here.

First, the *Initial Developer* grants a patent license:

> *...Under Patents Claims infringed by the making, using or selling of Original Code, to make, have made, use, practice, sell, and offer for sale, and/or otherwise dispose of the Original Code (or portions thereof). (MPL section 2.1[b].)*
>
> *Notwithstanding Section 2.1(b) above, no patent license is granted: 1) for code that You delete from the Original Code; 2) separate from the Original Code; or 3) for infringements caused by: i) the modification of the Original Code or ii) the combination of the Original Code with other software or devices. (MPL section 2.1[d].)*

These complex provisions draw important lines around patented intellectual property. They are the first explicit patent grant we've yet seen in open source licenses. The licenses I described earlier in this book contain at most implied patent grants. (If a license, like the GPL, is a *bare license*, then there may be no implied patent grant at all.) Implied patent grants are, at best, ambiguous.

Under the express provisions of the MPL, an *Initial Developer* licenses his or her patent claims to licensees of a specific embodiment of software, the *Original Code*, without limiting the *Initial Developer's* right to exclude others from making, using, or selling other embodiments in other software.

Patent Claims are potentially valuable even if the Initial Developer doesn't realize initially how his or her inventions will later be applied. The developer may discover that his or her claims cover very different applications from what was originally conceived, or such claims may cover applications

that combine technology contributed (or kept proprietary) by others. Thus, Patent Claims infringed by the making, using or selling of Original Code may find applications broader than just making, using, or selling the Original Code.

For example, a patent claim for a cut/paste function licensed under the MPL for use in specific Original Code that does word processing ("WP version 1") may have other valuable applications, such as in an email or graphics program. Consider an open source licensor, an Initial Developer, who distributes WP version 1. That licensor owns a patent that contains three valuable claims, which I will paraphrase very incompletely (and unprofessionally, were I actually writing patent claims) as follows:

(1) Software to perform a cut/paste function.

(2) The software of claim 1 for a word processor.

(3) The software of claim 1 for an email program.

The *Initial Developer* (i.e., the patent owner and MPL licensor) grants enough patent rights so licensees can make, use, or sell WP version 1, the *Original Code*. (See MPL section 2.1[b].) Licensees under the MPL thus obtain limited licenses to the *Initial Developer's* broad claim 1 and the narrower claim 2. (Claim 1 is broader than claim 2 because claim 2 only applies to word processors, but claim 1 applies to any cut/paste application.)

- Does the MPL patent license allow licensees to create and distribute a derivative word processor, WP version 2, which includes a cut/paste function? Probably not. The MPL patent license covers the *Original Code* only, *or portions thereof*.

(MPL section 2.1[b].) We'd have to examine WP version 2 to make sure that it contains at least the portions of *Original Code* that perform the cut/paste functions. If the cut/paste software in the *Original Code* is modified, the *Initial Developer*'s patent license doesn't cover it. (MPL section 2.1[d].)

- Can a licensee perform cut/paste functions in different word processors obtained from other licensors? Not if that software infringes the original licensor's broad claim 1 or narrower claim 2. The MPL patent license doesn't cover software separate from the *Original Code*. (MPL section 2.1[d].)
- Can a licensee perform cut/paste functions in email programs? No. The MPL patent license excludes claims that aren't infringed by making, using, or selling the *Original Code*. (MPL section 2.1[b].) The *Initial Developer*'s narrow claim 3 is excluded under the MPL license because an email claim is not infringed by the original word processing program.
- Can a licensee perform cut/paste functions in graphics programs? Not without a separate license to the *Initial Developer*'s broad claim 1. Notice that, in this example, the *Initial Developer* doesn't have a claim specifically covering graphics programs, but the cut/paste claim 1 is broad enough to apply to such new applications. Suppose that a *Contributor* invents a new graphics application for *Initial Developer*'s claim 1. Nothing prevents anyone from patenting separately an

> *improvement* on someone else's patent claim; he or she simply can't *practice* his or her improvement without a license to the broader claim. Two companies might thus create patent claims worth cross-licensing with each other, one a broad claim covering cut/paste, and the other a narrower claim covering cut/paste in graphics programs. Note that neither patent owner is required by the MPL to license his or her patent claims (the *Initial Developer's* claim 1 or the *Contributor's* graphics claim) to each other for open source or proprietary graphics programs.

In other words, the original MPL patent license applies only to claims 1 and 2, and only to a specific *Original Work* and to certain types of authorized *Modifications*.

As for a *Contributor*, this is that subsequent licensor's reciprocal patent license:

> *...Under Patent Claims infringed by the making, using, or selling of Modifications made by that Contributor either alone and/or in combination with its Contributor Version (or portions of such combination), to make, use, sell, offer for sale, have made, and/or otherwise dispose of: 1) Modifications made by that Contributor (or portions thereof); and 2) the combination of Modifications made by that Contributor with its Contributor Version (or portions of such combination). (MPL section 2.2[b].)*
>
> *Notwithstanding Section 2.2(b) above, no patent license is granted: 1) for any code that Contributor has deleted from the Contributor Version; 2) separate from the Contributor Version; 3) for infringements caused by: i) third party modifications of Contributor Version or ii) the combination of*

> *Modifications made by that Contributor with other software (except as part of the Contributor Version) or other devices; or 4) under Patent Claims infringed by Covered Code in the absence of Modifications made by that Contributor. (MPL section 2.2[d].)*
>
> *"Contributor Version" means the combination of the Original Code, prior Modifications used by a Contributor, and the Modifications made by that particular Contributor. (MPL section 1.2.)*

These provisions deal with *Modifications* submitted by *Contributors* who are licensees of the *Original Work*. Each *Contributor* grants a reciprocal license for his or her own patents to allow *Modifications* to be made, used, or sold either alone or in combination with the *Original Work*. So if a *Contributor* invents cut/paste software that works for graphics, or an entirely different invention (such as a new way of processing fonts), and includes it in his or her *Contribution*, that claim is reciprocally licensed to the *Initial Developer,* to all other *Contributors*, and to all subsequent *licensees*, under terms similar to the complex ones I've just described.

Furthermore, these provisions don't mean that the *Contributor* can automatically obtain a license to the *Initial Developer*'s claim 3 simply by creating a *Modification* that adds an email function; the *Initial Developer*, who has the right to license that patent claim, has specifically excluded it. (See MPL section 2.1[d].)

If you intend to become a *Contributor*, you may need an additional patent license from the *Initial Developer* or an earlier *Contributor* before you can make, use, or sell your *Modification*.

The MPL makes this explicit but, under the patent laws, the same issue exists under all the open source licenses, with their potential implied patent license grants, previously discussed in this book. Anyone planning to create improvements to open

source software must obtain licenses to any patent claims necessary to make, use, or sell those improvements. A patent grant from the licensor that would cover improvements is not implicit or explicit in any of the licenses I've discussed so far.

I like to think of such implied or explicit patent license restrictions as *field of use* restrictions; they limit the patent, sometimes in subtle ways, to use in specific fields or applications. How should we deal with *field of use* restrictions in open source licenses, where the copyright license provides unlimited freedom for licensees to create *derivative works* but the patent license does not?

The Free Software Guidelines, the Open Source Definition, and the Open Source Principles from Chapter 1 provide no guidance. They do not mention patents at all. The fundamental activities that open source deals with are copying, modification, and distribution. That's copyright law. What about patent rights: making, using, selling and offering for sale, and importing?

I believe the following is the only answer consistent with open source principles and with existing open source licenses:

> ***An open source license must grant enough patent rights to allow the licensee to make, use, sell, offer for sale, or import the open source work as distributed by its licensor. Any additional license rights for derivative works or other uses are at the option of the licensor.***

The first sentence, identifying the minimum scope of a patent grant in an open source license, probably describes how a court would decide anyway in the absence of an express patent grant in the license—at least for a *contract* although not for a *bare license*—because a right to copy software is usually meaningless without a right to make and use, and the right to distribute is meaningless without the right to sell.

The second sentence describes an option for increasing the scope of the patent grant and so doesn't belong in a mandatory Open Source Principle.

I ultimately decided to leave this patent principle out of the Open Source Principles entirely because several important already-approved open source licenses don't say anything at all about the scope of the patent grant. Otherwise we might have to declare some existing open source licenses incompatible with this patent principle, further confusing people about what *open source* really means.

Defending Against Patents

The MPL grants a limited license to the *Initial Developer*'s and *Contributors'* patents. But what is the MPL's response if a third party asserts its patents against an *Initial Developer* or *Contributor*?

The MPL handles this in various ways. First, the *Initial Developer* or any *Contributor* who learns about such a third party patent claim has an obligation to inform all subsequent licensees:

> *If Contributor has knowledge that a license under a third party's intellectual property rights is required to exercise the rights granted by such Contributor under Sections 2.1 or 2.2, Contributor must include a text file with the Source Code distribution titled "LEGAL" which describes the claim and the party making the claim in sufficient detail that a recipient will know whom to contact. If Contributor obtains such knowledge after the Modification is made available as described in Section 3.2, Contributor shall promptly modify the LEGAL file in all copies Contributor makes available thereafter and shall take other steps (such as notifying appro-*

> *priate mailing lists or newsgroups) reasonably calculated to inform those who received the Covered Code that new knowledge has been obtained. (MPL section 3.4[a].)*

A far more dramatic response is authorized by the MPL if someone actually files a patent infringement lawsuit against the *Initial Developer* or a *Contributor* (both of whom are now called a *Participant* and whose code is now called a *Contributor Version*). The provision is generally referred to as a *patent defense*; it can be found among the MPL's termination provisions in section 8.

The MPL's patent defense provision can be summarized this way: Participant will license you his or her Contributor Version—with the right to make free copies, prepare derivative works, and distribute—as long as you don't sue for patent infringement. But if you sue the Participant claiming that the Contributor Version itself infringes your patent, all copyright and patent licenses to you under the MPL for the Contributor Version are terminated. And, if you sue the Participant for any other patent infringement unrelated to the Contributor Version, all patent licenses to you under the MPL for any software are terminated.

The success of a *patent defense* depends on the perceived value of the *Contributor Version* to the third party patent owner. For important and valuable open source software, it may be more painful to the patent owner to forgo use of the software than to forgo some potential patent royalties. It at least forces a potential patent litigant to think carefully before he or she sues a Participant for infringement. Patent litigation is no longer risk-free.

Here's how the patent defense provision actually reads in the MPL:

> *If You initiate litigation by asserting a patent infringement claim ... against Participant ... alleging that:*
>
> *(a) such Participant's Contributor Version directly or indirectly infringes any patent, then any and all rights granted ... under Sections 2.1 and/or 2.2 of this License shall ... terminate prospectively....*
>
> *(b) any software, hardware, or device, other than such Participant's Contributor Version, directly or indirectly infringes any patent, then any rights granted to You by such Participant under Sections 2.1(b) and 2.2(b) are revoked.... (MPL section 8.2.)*

This is the first *patent defense* provision we have encountered in an open source license, and it has proven to be quite controversial and yet widely copied. There are several interesting variations on patent defense in other open source licenses; I will discuss some of these variations later.

Other Important MPL License Provisions

As I said, the MPL is the first of the industrial-strength open source licenses. It deals with issues that are typically the province of licensing and legal professionals. But because several of these are critically important to license enforcement, I introduce them here.

U.S. Government Rights

The MPL contains what must seem like cryptic instructions regarding U.S. government users of the *Covered Code*:

> *The Covered Code is a "commercial item," as that term is defined in 48 C.F.R. 2.101 (Oct. 1995), consisting of "commercial computer software" and "commercial computer software documentation," as such terms are used in 48 C.F.R. 12.212*

> *(Sept. 1995). Consistent with 48 C.F.R. 12.212 and 48 C.F.R. 227.7202-1 through 227.7202-4 (June 1995), all U.S. Government End Users acquire Covered Code with only those rights set forth herein. (MPL section 10.)*

The Code of Federal Regulations (C.F.R.) documents U.S. government policies and Chapter 48 of the C.F.R. contains Federal Acquisition Regulations. The relevant rules relating to patents, data, and copyrights are:

> *Commercial computer software or commercial computer software documentation shall be acquired under licenses customarily provided to the public to the extent such licenses are consistent with Federal law and otherwise satisfy the Government's needs. (48 C.F.R. 12.212.)*
>
> *Offerors and contractors shall not be required to ... relinquish control to, or otherwise provide, the Government rights to use, modify, reproduce, release, perform, display, or disclose commercial computer software or commercial computer software documentation. (48 C.F.R. 227.7202-1.)*
>
> *The Government shall have only the rights specified in the license under which the commercial computer software or commercial computer software documentation was obtained. (48 C.F.R. 227.7202-3.)*
>
> *A specific contract clause governing the Government's rights in commercial computer software or commercial computer software documentation is not prescribed. As required by 227.7202-3, the Government's rights to use, modify, reproduce, release, perform, display, or disclose computer software or computer software documentation shall be identified in a license agreement. (48 C.F.R. 227.7202-4.)*

Considering the broad scope of any open source license, under which the Government's rights—and everybody's

rights—to use, modify, reproduce, release, perform, display, or disclose computer software is unquestioned, it is hard to imagine why open source licenses would need a U.S. Government Rights provision like the one in the MPL. The United States government—just like everybody else—is being given a license to free software. What more or less do they need?

Representations

> *Contributor represents that, except as disclosed pursuant to Section 3.4(a) above, Contributor believes that Contributor's Modifications are Contributor's original creation(s) and/or Contributor has sufficient rights to grant the rights conveyed by this License. (MPL section 3.4[c].)*

The MPL is the first license to assure licensees that *Modifications* are original to the *Contributors* who submit them or are being distributed under the authority of the original author.

This concept will appear as a *Warranty of Provenance* in the OSL/AFL licenses described in Chapter 9.

Jurisdiction and Venue

In the event of a dispute about the MPL, California law applies. As specified in the license, any litigation will take place in the federal courts of the Northern District of California, with venue in Santa Clara, California. (MPL section 11.)

I will discuss jurisdiction and venue, as well as governing law, in Chapter 12.

Attorneys' Fees and Costs

In the event of a dispute about the MPL, the losing party in court must pay *reasonable* attorneys' fees and costs. (MPL section 11.) What is *reasonable* is left to a court to decide.

Software Is Not Goods

I noted very early in this book that it is important to distinguish personal property rights in the copy of the software acquired in a store, and property rights in the intellectual property embodied in the software. Software licensed under the MPL is specifically intended *not* to be subject to laws intended for the sale and distribution of goods in international commerce.

> *The application of the United Nations Convention on Contracts for the International Sale of Goods is expressly excluded. (MPL section 11.)*

The most important reason for this provision is to ensure that international laws concerning implied warranties won't apply to this software. As the MPL and other open source licenses remind everyone, the software is provided on an "AS-IS" basis. (MPL section 7.)

This provision may not be enforceable in all jurisdictions.

Multiple-Licensed Code

I haven't yet explained why many open source licensors find it useful to license their software under more than one license. That topic will come in Chapter 11 when I discuss dual licensing models.

The MPL makes it clear, however, that the *Initial Developer* may designate its software as being available under multiple licenses, and may specify which license, besides the MPL, is allowed. (Frequently the second license is the GPL.) Note that a *Contributor* under the MPL cannot, independently, elect to use a different license for his or her *Modifications*. Only the Initial Developer makes that choice. This point will be discussed in Chapter 10 when I address the problem of *relicensing* open source software.

Other Corporate Licenses

Open source software has been adopted by many of the world's largest software companies. While most have adopted one or another of the licenses already discussed in this book (either the GPL or one of the academic licenses), some of them also now distribute their own open source software under their own corporate licenses. Open Source Initiative now lists licenses from a number of major companies including Apple, Lucent, IBM, Intel, Nokia, Real Networks, Ricoh, Sun, and Sybase. (See *www.opensource.org*.)

Each of those licenses puts a spin on one or another licensing technique already described in this book. Examining each of them in turn would be unproductive. Every one of those licenses satisfies the Open Source Principles listed in Chapter 1, although they sometimes do it in unusual ways.

The specific provisions of each license matters, particularly if you intend to create and distribute derivative works. If you use open source software from those companies under their licenses, I suggest that you consult an attorney to make sure you honor your obligations.

For the most part, those licenses are intended for use by the company that placed its name on it. None of them is an effective template that can be used by licensors generally.

There are three important exceptions. The first, the Common Public License (CPL) written by attorneys at IBM, is described in Chapter 8. Two other template licenses which I wrote, the Open Software License (OSL) and the Academic Free License (AFL), are described in Chapter 9.