4

Taxonomy of Licenses

What Is a License?

I've used the word *license* quite loosely in the preceding chapters, waiting for an opportune time to explain that word from a legal perspective. In one sense, a license is a permission to do something. The government issues licenses, such as a license to drive a vehicle on the public right of way or a license to run a business, pursuant to laws regulating such activities. The government tells you that you may not drive a car or engage in business without an appropriate license. You are required to obey the traffic laws and the laws regulating businesses, although the license you bought has nothing to do with those obligations. If you exceed the speed limit or if you engage in a fraudulent business practice, you can be penalized even if you didn't bother to get an appropriate license.

An owner of a private property right can grant licenses to allow others to exercise property rights that otherwise would be exclusive to the property owner. For example, the owner of beachfront property can license a telescope club to pass onto the beach to witness a solar eclipse. (There are subtle differences between this kind of license and an easement that grants access to real property, about which nothing more will be said in this book.) Such licenses can be limited as to time. They may grant rights only to specific people or to the public as a whole.

In this book, the term *license* is used to describe the legal way a copyright and patent owner grants permission to others to use his intellectual property.

An *open source license* is the way a copyright and patent owner grants permission to others to use his intellectual property in such a way that *software freedom* is protected for all.

A *proprietary license* is the way a copyright or patent owner grants permission to others to use his intellectual property in a restricted way, through secrecy or other limitations, so that software freedom is not protected.

The word *proprietary* is often confused with the word *commercial*. But a *commercial license*—which is merely a term used to describe a license used in commerce—can be either open source or proprietary.

Licenses can be express or implied. An express license is typically a written document that is reviewed and agreed to by the owner of the licensed property (the *licensor*) and by the receiver of the license grant (the *licensee*). All of the licenses described in this book contain at least some express written terms and conditions.

A license may also be implied by the kind of license being granted, by the conduct of the licensor, or by the licensor's apparent refusal to exercise its exclusive rights to the licensed property. In one very important example, some open source licenses say nothing about a grant of patent license, leaving the patent license to implication.

Be careful about implied licenses. An implied license is necessarily vague and incomplete. The terms and conditions of an implied license may not be clear to either the licensor or the

Rosen_ch04 Page 53 Tuesday, June 22, 2004 7:39 PM

licensee. Reliance on an implied license is particularly risky when important property interests are at stake.

Bare Licenses

I now address a topic that is a kind of Heisenberg Uncertainty Principle of open source: Are open source licenses bare licenses or are they contracts? The answer to this question depends on how you look and what you're trying to measure. Open source licenses, it turns out, can be both bare licenses and contracts. Adding to the confusion, the parties to open source licenses are typically referred to as *licensor* and *licensee* regardless of whether the licenses are bare licenses or contracts.

Among the examples I cited in the previous section was one about drivers' licenses. A driver's license is issued by a government agency, but it does not constitute an agreement of any sort between the driver and the agency. There is no contract; the driver's license is merely a permission slip. The licensor has made no promises and neither has the licensee.

Private parties also can grant licenses. In the software licensing context this is what we mean:

> *Bare license: A grant by the holder of a copyright or patent to another of any of the rights embodied in the copyright or patent short of an assignment of all rights. (Merriam-Webster's Dictionary of Law 1996.)*

It is possible for a copyright owner to grant a license to copy, modify, and distribute software without signing a contract between the parties. The argument goes like this: Since those exclusive rights cannot be exercised without the permission of the copyright owner, a licensee must either obey the terms of the license or not exercise the rights. Anything else is copyright or patent infringement.

Here is how one open source license, the GPL, expresses this point:

> *You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing, or modifying the Program or works based on it. (GPL section 5.)*

This reference to *acceptance* in the GPL involves a concept from contract law. Quite simply, a contract cannot be formed unless there is both an offer (from the licensor) and acceptance (by a licensee). Licensees are not required to accept the GPL, and if they don't accept, a contract is not formed. But a bare license has been granted—a bare license that ceases to exist if the terms and conditions are not obeyed.

The law governing an open source license in the absence of a contract is the Copyright Act, Title 17, of the U.S. Code, the equivalent laws of other countries, and international copyright treaties. To the extent that patent rights are implicated, the law governing the license is the Patent Act, Title 35, of the U.S. Code, the equivalent laws of other countries, and international patent treaties.

Those laws forbid anyone from exercising the exclusive rights of a copyright or patent owner without a license. If such a person doesn't have a license, he is an infringer subject to substantial penalties. (See Chapter 12 for a discussion of open source litigation.)

One problem with treating open source licenses as bare licenses is that intellectual property law does not say much about how to interpret license terms. Attorneys and courts are familiar with licenses that are contracts and they regularly apply the well-developed law of contracts to handle issues of license interpretation. In the absence of contract law, there is no ready framework for license language interpretation.

This practical interpretation problem can take many forms. When a license like the GPL doesn't even demand acceptance, can a licensor assume that licensees have agreed to all of those terms? What about terms that are inconsistent with consumer protection laws such as certain warranty disclaimers? What about terms in a license that are inconsistent with the definitions of terms of art in copyright law, such as derivative work or distribution? If there is no express agreement by the parties to a common set of terms and conditions, can the licensor's interpretation of the terms and conditions be enforced against the licensee? Did the licensee accept the differing definitions?

There is no body of cases and statutes to help us answer those questions. In the absence of a contract, the terms and conditions of a bare license may be subject to varying court interpretations around the world. Some legal scholars even argue that terms and conditions of bare licenses like the GPL are completely unenforceable, although the legitimacy of the GPL has never been tested in any court. Neither have any other open source licenses. This vague uncertainty hovering over bare licenses like the GPL has not been much of an obstacle to the adoption of GPL-licensed software, but it is unpleasant for attorneys nonetheless.

Another practical problem with bare copyright licenses is that only the owners of copyrights and patents can enforce those copyrights and patents in court. The cause of action for a refusal to comply with the terms and conditions of a bare copyright or patent license is just infringement rather than also breach of contract. This causes open source distributors to concern themselves with "who owns the copyrights or patents," rather than "who licensed this software." (This topic is also discussed more fully in Chapter 12.)

A third problem with bare licenses is that they may be revocable by the licensor. Specifically, *a license not coupled with an interest may be revoked.* The term *interest* in this context usually means the payment of some royalty or license fee, but there are other more complicated ways to satisfy the interest requirement. For example, a licensee can demonstrate that he or she has paid some consideration–a contract law term not found in copyright or patent law–in order to avoid revocation. Or a licensee may claim that he or she relied on the software licensed under an open source license and now is dependent upon that software, but this contract law concept, called promissory estoppel, is both difficult to prove and unreliable in court tests. (The concepts of *consideration* and *promissory estoppel* are explained more fully in the next section.) Unless the courts allow us to apply these contract law principles to a license, we are faced with a bare license that is revocable.

Most of those issues about bare licenses have never been addressed directly in a court so lawyers have no good way to predict how they will ultimately be answered. In the absence of a court decision interpreting bare open source copyright licenses, distributors of software under such licenses should ask their attorneys whether they have adequate protection.

In my opinion, it is safer for a licensor and his licensees to enter into enforceable contracts. That usually doesn't require any changes to the license text; it only requires that the license be offered and accepted as a contract, and that there be an understanding between the parties about the consideration paid for the license.

Licenses as Contracts

Read in a different light, open source licenses contain promises, just like ordinary contracts. In effect, each licensor promises, subject to certain terms and conditions, not to interfere with licensees who copy, modify, distribute, make, use, and sell open source software embodying the licensor's intellectual property. Licensees rely on those promises when they adopt open source software to do useful things.

Many open source licenses are designed as contracts.

> *A contract is a promise or set of promises for breach of which the law gives a remedy, or the performance of which the law in some way recognizes as a duty. (Restatement, Second, Contracts § 3.)*
>
> *A promise is a manifestation of intent to act or refrain from acting in a specified way, so made as to justify a promisee in understanding that a commitment has been made. (Restatement, Second, Contracts § 2.)*

I'll discuss later in this book the specific promises made (express and implied) in open source licenses. In particular, there are software licenses called *unilateral contracts*, in which only the licensor makes promises, and other licenses called *bilateral contracts*, in which both parties make promises. Most open source licenses are unilateral in intent. (Even lawyers who draft licenses are sometimes confused by these concepts; you will occasionally find terms of art, such as "licensee agrees" promissory language appropriate for *bilateral* contracts, in otherwise *unilateral* contracts.) For now, it is important only to identify the differences between a bare license and a contract.

Contract law, unlike copyright and patent law, provides procedures and rules for license interpretation and enforcement. Contract law, in the published court decisions and in the statutes adopted by legislatures around the world, addresses almost every possible term or condition a lawyer could dream up for a contract. Contract law specifies how contracts are to be formed, how they are to be interpreted, how they are to be enforced, and the remedies for breach. In many situations, where a license is silent about a particular term or condition, contract law even provides default “fill-in” provisions.

Some suggest that since contract law varies around the world, open source contributors and distributors should rely exclusively on consistent copyright and patent law for their licenses. But the varieties of contract law are exaggerated, as are the similarities of copyright and patent law around the world. The global requirement for consistency of commercial transactions—a requirement of the capitalist market system—helps ensure that contracts are interpreted in much the same way around the world. Meanwhile copyright law is *not* consistent; the courts around the world, for example, don’t agree on what constitutes a derivative work of software. That is why it is sometimes better for an open source contract to define the term *derivative work* than to have a bare license simply use that term of art as if it had a consistent meaning worldwide.

Unlike a bare license, a contract can be enforced by a licensor even if he doesn’t own the underlying copyrights and patents. This means that a distributor of software can enforce his contract against his licensees without needing the approval of the copyright and patent owner(s) to do so. For open source software containing original software contributed by programmers worldwide, it can be particularly important for a distributor to be able to enforce his licenses even without owning the underlying patents or copyrights.

Finally, the generally accepted rule that *the contract is the law* encourages us to create complete licenses that state the terms and conditions as clearly as we want. We don't have to rely on vague interpretations of copyright or patent law since we can write the law-of-the-contract exactly as we want it to be enforced. For example, later in this book I will describe two recent open source licenses, the Academic Free License (AFL) and the Open Software License (OSL), that specify in contract form and in clear and precise terms the rules for open source licensing. Those licenses—one an academic license and the other a reciprocal license, but otherwise identical—are intended to be enforceable under both contract and copyright law.

The main difference between a bare license and a contract is in the way the relationship between licensor and licensee is formed. To create a contract, there must be an offer and acceptance, and there must be consideration. I will describe these three elements in turn. (In first-year contract law courses, these elements are often referred to as the *legs of a stool;* a contract is the seat of the stool; it will fall if any of the legs—offer, acceptance, or consideration—fails.)

None of these three elements is needed for a bare license.

Offer

An *offer* is fairly simple in the software licensing context.

> *An offer is a manifestation of willingness to enter into a bargain so made as to justify another person in understanding that his assent to that bargain is invited and will conclude it. (Restatement, Second, Contracts § 24.)*

In an open source license, the licensor offers to allow licensees to copy, modify, and distribute the licensed software for any purpose whatsoever in accordance with the Open Source Principles in Chapter 1.

The appropriate manifestation of willingness required for an offer can be (and often is) expressed by posting the software on some Internet portal like SourceForge or on a public website in such a way that all prospective licensees will be able to retrieve the software under the terms of the license. Open source distributors offer licenses to everyone.

Acceptance

The offer empowers the licensee to create a contract by his acceptance. The second step in forming a contract, then, is for the licensee to accept it. He must *intend* to accept it.

Traditionally, a signed written agreement is evidence of both offer and acceptance, but that is no longer practical with the mass marketing of software. The most typical way to obtain acceptance of a software license is to require licensees to express their assent in a positive way, such as by making a purchaser of boxed software open an inner package that boldly announces the presence of the license (known as *shrink-wrap*), or by making someone who downloads software click on an "I ACCEPT" button on a website (known as *click-wrap*). Many courts around the world now agree that clicking on "I ACCEPT" or tearing the shrink-wrap is ample evidence that the licensee accepted the contract.

The law doesn't require shrink-wrap or click-wrap. Indeed, for many forms of software distribution and installation, neither of those specific techniques is appropriate. Any acceptance procedure that ensures an explicit manifestation of assent is usually sufficient. Even that is difficult to accomplish when open source software is merely posted and distributed on the Internet. So it is important to understand the implications of not obtaining an *explicit manifestation of assent* up front. There are three alternative situations:

Rosen_ch04 Page 61 Tuesday, June 22, 2004 7:39 PM

- Both parties can later affirm that they intended to form a contract and agree to abide by its terms and conditions. That subsequent stipulation suffices to prove acceptance. (The courts won't care as long as the parties agree among themselves.)

- The licensor wants out of the contract: In the case of a unilateral contract (such as almost all the open source licenses in this book) in which the licensor is the only one making promises, the subsequent testimony of the licensee that he intended to accept the contract and that he acted in reliance on it is usually sufficient evidence of acceptance even if the licensor now wants out of the contract.

- The licensee wants out of the contract: As long as the licensor wants to enforce the contract, the licensor has the burden of proving that a contract was formed. This situation demonstrates why licensors should demand an explicit manifestation of assent that they can introduce as evidence if necessary.

Consideration

The third requirement for contract formation, consideration, is often the most complicated.

> *(1) To constitute consideration, a performance or a return promise must be bargained for.*
>
> *(2) A performance or return promise is bargained for if it is sought by the promisor in exchange for his promise and is given by the promisee in exchange for that promise.*

> *(3) The performance may consist of (a) an act other than a promise, or (b) a forbearance, or (c) the creation, modification, or destruction of a legal relation. (Restatement, Second, Contracts, § 71.)*
>
> *If the requirement of consideration is met, there is no additional requirement of (a) a gain, advantage, or benefit to the promisor or a loss, disadvantage, or detriment to the promisee; or (b) equivalence in the values exchanged; or (c) mutuality of obligation. (Restatement, Second, Contracts, § 79.)*

Taken together, these two legal principles from the Restatement prevent the enforcement of a *gift*, which may have both offer and acceptance but lacks the element of consideration. Section 79 in particular makes it clear that the value of the consideration, while it can't be zero, doesn't need to be very large at all. Early legal scholars made the point that a peppercorn could be sufficient consideration for a contract.

To cut to the chase, I'll refer to the following Simple License:

> *The copyright owner of this software hereby licenses it to you for any purpose whatsoever.*

This is, of course, a bare license. Like any bare license, it is enforceable by the copyright owner under copyright law and can be revoked by the licensor at any time.

Assume, now, that we want this Simple License to be treated as a contract so that it can be enforced under contract law and so that it cannot be revoked. Assume also that we have satisfied the procedural requirements for offer and acceptance. Where can we find consideration in the language of the Simple License?

Laws in some jurisdictions provide that specified types of promises are enforceable without consideration. This is usually restricted to certain commercial transactions and written contracts. While it is not common now, the growth of the open source software industry may eventually demand that, by statute, the grant of a written license to computer software in commercial settings creates an enforceable contract between licensor and licensee even in the absence of consideration. Without such a legal exception, however, we must find consideration or we don't have a contract.

Perhaps we can look deeper into the Simple License to find consideration, even though *consideration* isn't among the express words of the license. Consideration might be implied.

The licensor's detriment is an implied result of copyright law. The licensor has licensed the otherwise exclusive rights under copyright, and as to that licensor, forbearance to enforce those exclusive rights is detriment (e.g., consideration) enough.

What about consideration or detriment by the licensee?

The easiest way for the licensee to ensure that the Simple License can be enforced as a contract is if he pays a royalty or license fee for the software to be used, copied, modified, and distributed. It needn't be much, and perhaps a penny is sufficient, but there must be consideration by the licensee or there is no contract. (That is not contrary to the Open Source Principles; some open source software is sold in stores.) That demand for payment needn't be expressed in the Simple License itself, because although consideration is an element of contract formation, it is not necessarily a part of the contract itself. Consideration may be obtained by demanding a license fee before allowing download of open source software. Of course, licensors should avoid sham consideration—such as a penny—that might convince a court that a gift rather than a contract was intended.

Many customers obtain their open source software from established commercial enterprises either combined with hardware and services or as part of a comprehensive support package. Those associated agreements often establish the element of consideration that is required for treating the license itself as a contract.

But ultimately, the issue of price is irrelevant for most open source software. Most is available truly free of charge for those who want it. Not even a penny is demanded for its download. Where can we find consideration by a licensee in an open source license that otherwise promises the free use of software—at zero price—and allows copies and derivative works to be distributed without payment of royalties? (See Open Source Principles # 1, 2 and 3.)

This question becomes even more confusing when we realize that open source licenses are almost always written as unilateral contracts in which only the licensor has made promises. At no time has the licensee been requested to bind him- or herself to do anything, and even if the licensee starts to use the software that licensee is not bound to continue to do so. A court may find the necessary detriment to the licensee, and thus the necessary consideration, in the very act of using, copying, modifying, and distributing the software. This is the basis of the contract law doctrine of *promissory estoppel*, in which *detrimental reliance* becomes a substitute for consideration. The law of contracts describes it as follows:

> *A promise which the promisor should reasonably expect to induce action or forbearance on the part of the promisee or a third person and which does induce such action or forbearance is binding if injustice can be avoided only by enforcement of the promise. The remedy granted for breach may be limited as justice requires. (Restatement, Second, Contracts, § 90.)*

A court may find detrimental reliance by licensees who have accepted open source software for use in the infrastructure of the modern economy. It is inconceivable to me, for example, that licensors of Linux, or Apache, or any of the other major open source software packages, would be allowed to revoke their licenses for lack of consideration. But it remains to be seen whether promissory estoppel will generally serve as a substitute for consideration in open source licensing. It has never been tested in court.

Just because there is uncertainty about the element of consideration shouldn't lead us to ignore the other two elements of contract formation, offer and acceptance. A court is unlikely to find promissory estoppel when licensors haven't even made the effort to offer clear promises in the first place and to get them accepted.

If open source licenses are to be treated as contracts, all three elements of contract formation should be satisfied wherever possible.

Failure of Offer, Acceptance, or Consideration

Of all the licenses described in this book, only the GPL makes the explicit point that it wants nothing of *acceptance* or *consideration*:

> *You are not required to accept this License, since you have not signed it. (GPL section 5.)*
>
> *You must cause any work that you distribute or publish ... to be licensed as a whole at no charge to all third parties under the terms of this License. (Underline added; GPL section 2[b].)*

The GPL authors intend that it not be treated as a contract. I will say much more about this license and these two provisions in Chapter 6. For now, I simply point out that GPL licensors are in essentially the same situation as other open source licensors who cannot prove offer, acceptance, or consideration. There is no contract.

What is left? Even if the contract fails, a bare license remains, and that license can be enforced under copyright law—with all the limitations on such enforcement actions described earlier—or it can be revoked.

Here is how the Open Software License and the Academic Free License make this legal point:

> *Any use of the Original Work outside the scope of this License or after its termination shall be subject to the requirements and penalties of the U.S. Copyright Act, 17 U.S.C. § 101 et seq., the equivalent laws of other countries, and international treaty. This section shall survive the termination of this License. (OSL/AFL section 11.)*

Even if this provision isn't explicit in all open source licenses, that's probably the way the law will treat the situation anyway.

Also note that licensees have little to gain by denying the existence of a contract unless they're willing to have their licenses revoked, and licensors almost always want their contracts enforced. Litigation about contract formation issues probably won't arise in commercially relevant situations.

Patent Licenses

There is an entire breed of specialized licenses that are used for patents. Patent owners license their patent rights to other companies, authorizing the licensees to make, use, sell or offer for sale, or import products embodying the claims of the patent. Rarely are such patent licenses unlimited. Instead, we typically see limitations for specific fields of use (e.g., a semiconductor patent licensed only for making disk drive heads), for specific products (e.g., a browser patent licensed only for a particular operating system), or for specific markets and geographic regions (e.g., a telephone system patent licensed only for products sold in the European Community).

To be compatible with an open source license, a patent license necessary to make, use, or sell the software under license must not prevent the creation of derivative works or prohibit use anywhere in the world. (See Open Source Principles #1 and 3.)

Patent licenses often require payment of royalties to the patent owner. Such licenses may be incompatible with open source licenses if they require licensees or sublicensees to pay for the right to make and distribute copies or derivative works. (See Open Source Principles #2 and 3.) Some *paid-up* patent licenses, which require a single up-front payment for all patent rights, can be consistent with open source software. But it is difficult to find an angel to invest significant money in a paid-up patent license where those costs cannot be passed on to downstream licensees.

Large companies with extensive patent portfolios often negotiate cross-licenses with other companies. Each party to the license agrees to allow the other to make, use, sell or offer for sale, or import products embodying claims in the licensed portfolios. Such patent licenses are compatible with open source licenses as long as the software licensor has rights, under the cross-license, to allow downstream open source–compatible patent licensing.

It is difficult in a book like this to say much of value about stand-alone patent licenses. Software is not licensed that way because software is inevitably both copyrightable and patentable. A software license always has a copyright component. Where stand-alone patent licenses do become important to open source is in the context of open standards that are intended to be implemented in software. These specialized patent licenses for open standards are discussed in Chapter 13.

For now, I'm going to focus on the patent license grants contained within open source licenses themselves. Such licenses convey sufficient patent rights to make, use, sell or offer for sale, or import the specific software in ways consistent with the Open Source Principles. These patent licenses are *implied* in some open source licenses, *expressed* in others. Patent license terms differ subtly among open source licenses. I will point this out when I introduce each license.

Template Licenses

Since a software license is a specific contract between two parties, a specific licensor and a specific licensee, there are literally millions of such licenses in effect today. Fortunately, many of those licenses have very similar wording. Rather than negotiate one agreement at a time, many software companies use fill-in-the-blank agreements drafted by their attorneys, defining the licensor and licensee as, for example, Company X and Company Y, respectively, but otherwise the same. In such ways, large companies often license large proprietary software packages using standard terms and conditions. It would be a waste of time to redraft and negotiate every license agreement afresh.

For mass marketed software, software licenses are even more generalized, defining the licensor and licensee as Company X and Licensee, respectively, where *Licensee* is defined generally as "the person or company exercising rights under this license," or words to that effect.

Open source software licenses sometimes add yet another level of generality. They don't specifically name Company X as the licensor, instead defining *Licensor* as "the person or company granting rights under this license," or words to that effect. That can allow a single form of license to be used without modification for many licensors and many licensees. These generalized licenses are sometimes called *license templates*.

Often more than the names of the licensor and licensee are replaceable in the template. Other template fields can be the name of the software, the copyright notice, or even important matters such as jurisdiction and governing law.

At the end of the day, however, it is essential to tie together a specific piece of software, a specific licensor, and a specific licensee, because it is those three pieces of information that determine what license terms apply to the specific parties doing the licensing. A license template without the blanks filled in is not a complete license.

As I discuss various licenses in this book, I will identify the ways, if any, that they serve as license templates.

Types of Open Source Licenses

With as difficult a concept as *software freedom* to contend with, it is not surprising that many licenses have been proposed to implement it. As of this writing, over fifty approved open source licenses are listed aty *www.opensource.org*. Understanding those licenses would be impossible without a licensing taxonomy, a way of organizing those licenses into appropriate categories.

Licenses generally fall into these categories:

- *Academic licenses*, so named because such licenses were originally created by academic institutions to distribute their software to the public, allow the software to be used for any purpose whatsoever with no obligation on the part of the licensee to distribute the source code of derivative works. The Berkeley Software Distribution (BSD) license used by the University of California to distribute its software is the archetypal academic license. Academic licenses create a public commons of free software, and anyone can take such software for any purpose—including for creating proprietary collective and derivative works—without having to add anything back to that commons.

- *Reciprocal licenses* also allow software to be used for any purpose whatsoever, but they require the distributors of derivative works to distribute those works under the same license, including the requirement that the source code of those derivative works be published. The GPL license, written by Richard Stallman and Eben Moglen at the Free Software Foundation, is the archetypal reciprocal license. Anyone who creates and distributes a derivative work of a work licensed under a reciprocal license must, in turn, license that derivative work under the same license. Reciprocal licenses, like academic licenses, contribute software into a public commons of free software, but they mandate that derivative works also be placed in that same commons.

- *Standards licenses* are designed primarily for ensuring that industry standard software and documentation be available to all for implementation of standard products. These licenses sometimes require that any differences from the industry standard be published as a reference implementation so that the standard may evolve if necessary.

- *Content licenses* ensure that copyrightable subject matter other than software, such as music, art, film, literary works, and the like, be available to all for any purpose whatsoever. These licenses are discussed more fully on the Creative Commons website at *www.creativecommons.org*. While the Creative Commons goals are not directly related to *software freedom*, there are many similarities of objective. A few of the software licenses discussed in this book, in particular the Academic Free License (AFL) and the Open Software License (OSL), are appropriate for use with content as well as software, as will be explained in due course.

Over the last few years, many organizations and companies have embraced open source software. In the process, they have written many open source licenses that are subtle variants on the academic and reciprocal themes. Those licenses are submitted to Open Source Initiative for review of compatibility with the Open Source Definition and approval as an open source license. There are already over fifty OSI-approved open source licenses.

All of the licenses discussed in this book are published at the website run by Open Source Initiative, *www.opensource.org*. Only approved licenses are listed. Software distributed under any of those licenses is OSI Certified open source software.

Open Source Initiative created a certification mark for licensors to display on open source software. As long as an OSI-approved license is used for distribution of the software, such open source software can be marketed with this certification mark:

OSI certified™