10

Choosing an Open Source License

How Licenses Are Chosen

I have been involved with the open source community long enough to recognize that decisions in projects about licensing strategy are almost always thoughtfully and carefully considered. Indeed, I learned far more about open source licensing from listening to those online licensing discussions than I ever learned about this topic in law school. The leaders of open source projects are knowledgeable about the law, committed to the principles of open source, and determined to create a commons of free software available to all. And so they write and choose licenses with intelligence and passion.

For many commercial companies, the discussion of which license to use, at least in the early stages, often centers on one or both of the following issues:

1. How can we make money from distributing this software under an open source license? In essence, can our license help us sell free software?
2. How can we prevent others from making money unfairly from our open source software? This is the so-called free-rider issue, where licensees reap

Rosen_ch10 Page 230 Wednesday, June 23, 2004 10:04 AM

all the benefits of others' work with no return obligations.

These questions are addressed in reverse order in the next two sections of this chapter.

The Free-Rider Problem

The second question posed above is actually the easier one. Under Open Source Principles # 2 and 3, it is impossible to completely prevent free-riders in open source. All licensees are free to copy and create derivative works without payment of royalties to the licensor, and so a licensee can make as many copies of such software as possible without rewarding the licensor with even a peppercorn as payment.

If it is important to discourage free-riders who create and distribute derivative works, then a reciprocal license is often more effective than an academic license. At least with reciprocal licenses, everyone is a free-rider of everyone else's distributed derivative works, because that software is licensed under the same license. The pain of the free-rider problem is equally shared by all distributors of derivative works, not just by the original licensor, under reciprocal licenses.

But whether a licensor chooses to distribute under an academic or a reciprocal license, the growing commons of open source software that generally results from open source licensing is believed by most in the open source community to be sufficient reward for allowing everyone to be free-riders.

If after considering open source models you still want to prevent free-riders, you should consider adopting one of the non–open source licenses described in Chapter 11, or try instead to make money with a proprietary software distribution model.

Making Money from Open Source

The first question I posed above ("How do I make money at this?") is far more difficult to answer. Broad copyright and patent licenses such as those described in this book are certainly not consistent with business models that rely upon selling software at high per-copy prices. Anyone can become an open source distributor and compete on price. This inevitably drives the per-copy price downward toward its marginal cost of production and distribution.

But licensors can make money on what the open source license *doesn't grant*. For this reason, it is often more rewarding to consider the *exclusions from license* rather than the open source *grants of license* when looking for opportunities for profit.

The most important exclusion is trademark or brand identity. Trademarks are excluded from all open source licenses, either explicitly or implicitly. Under the law, for the licensor to do otherwise would risk loss of his or her trademarks. It would result in a dilution of the licensor's trademarks to the point that consumers wouldn't know what specific software it represents.

Despite their protestations that quality matters most, companies and individuals usually acquire software not by function but by reputation. Trademarks are thus very important factors in consumer decisions. Given consumer behavior, it is no surprise that Linux and Windows are valued trademarks in the software marketplace. By marketing software under a trademark, the licensor can sell perceived value even though the underlying software might be available elsewhere for free without the trademark.

As to what steps to take to turn trademarks into profit, that is an exercise best left to discussions between your business strategists and your own intellectual property attorney. Suffice it to say that customers are often willing to pay for brand-name software, particularly if it comes with support and other benefits. Most open source licenses don't adversely affect that business opportunity at all.

Also excluded from several open source licenses, for many of the same reasons, are the names and reputations of the licensors. Even though they grant licenses to their software, licensors can protect their names and reputations for personal profit. Many individual contributors whose names adorn copyright notices in valuable software are making a good living because their professional reputations were enhanced by those contributions. They essentially sell themselves and their expertise, rather than their software.

Most warranties are also excluded from the open source licenses in this book. A separate business can be made from selling such warranties—as well as other support and installation services—separate from the software itself.

Finally, the licensor of open source software is always free to license his or her software *also* under other terms and conditions. This means that a prospective licensee who prefers to accept the software under a different license than an open source one—and who is willing to pay for that "advantage"—may contact the licensor to determine if the software is also available under a different license. Chapter 11 discusses some examples of dual licensing.

In-Licensing

Consider first the process of software licensing from the perspective of the recipient of the license, the *licensee*. Of course, a licensee doesn't choose the license; open source software is usually offered on the licensor's terms, without negotiation. In some cases, even the licensor may actually have no choice in the license, as when an open source project uses a reciprocal license, as in the GPL, MPL, or OSL that mandates a license for modifications.

So since you probably can't negotiate the license, the main issues that should concern you if you in-license software is whether the terms and conditions of the license being offered are acceptable given your business goals.

These are the typical considerations:

- Do you understand the terms and conditions of the license, or are there ambiguities and uncertainties that might affect license interpretation by reasonable parties or by a court?

- Are you intending to create and distribute derivative works of the software? If so, can you accept the reciprocity obligation of the license? Are you willing to distribute your derivative works under the same license? Are you satisfied with the license's definition of derivative works?

- Does the license grant you sufficient patent rights to create derivative works? If not, what other patent licenses will you need to make, use, sell, offer to sell, or import derivative work software?

- Does the licensor actually provide source code? Will you actually ever need it?

- Will you need any additional rights not included in the license, such as the right to apply trademarks to your goods?
- Are you prepared to honor license conditions relating to copyright and other notices?
- Do you need broader warranty protection than is offered under the license? If you do, is additional warranty protection offered at a price?
- Do you accept the limitations of liability under which the software is offered?
- Are you prepared to accept the jurisdiction, venue, and governing law provisions of the license? If you ever have to litigate this license, where and how would it be done?
- Are you prepared to accept the license termination provisions? Assuming you are going to invest in adopting and using the software in important ways, what is the chance that your license to the software may terminate?

Notice that there is one consideration that has already been dealt with if you accept an approved license listed on the Open Source Initiative website, *www.opensource.org*: You may be certain that the software license meets the Open Source Definition and the five Open Source Principles listed in Chapter 1 of this book. You can copy, create derivative works, distribute, make, use, and sell the open source software that you inlicense.

Rosen_ch10 Page 235 Wednesday, June 23, 2004 10:04 AM

Out-Licensing

Licensors decide what license to use for their open source software. If at all possible, licensors should use an existing template license. Please don't invent your own. The open source community is not seeking new licenses to analyze and interpret.

The proliferation of open source licenses creates a serious problem: It risks additional fragmentation of the public commons of free software. While software under some academic licenses can be combined without restriction, combining software under different reciprocal licenses—particularly the more complex reciprocal licenses used by large companies—requires that lawyers or skilled licensing professionals review each of the licenses for incompatibilities. Even where the differences between licenses are trivial, such as their designation of governing law, a combinatorial analysis of open source licenses rapidly becomes a nontrivial exercise. For example, it is a nontrivial exercise to determine whether a work that combines two separately licensed programs requires a file-by-file, MPL-like comparison or the more general *work based on the Program* derivative work test of the GPL. I say more about this problem later in this chapter.

Without exception, leaders of the open source community discourage the submission of "yet another license." Any software company deciding to distribute its software under an open source license is fervently encouraged to select among the existing licenses rather than to create a new one.

- As before, the key licensing factor is whether to use a reciprocal or an academic license. As a licensor, do you want to be able to benefit from improvements made by others? Do you want derivative works created by your licensees to be distributed under the same license so that you can incorporate their improvements into your own software?

- If a reciprocal license is desirable, you should consider the scope of the reciprocity obligation. Licenses like the GPL contain vague provisions about derivative and collective works; some licensors prefer that ambiguity because it results in more software licensee contributions licensed under the GPL. Licenses like the MPL have a more narrow definition of derivative works, requiring only *files* that are changed to be distributed under the MPL; this can reduce resistance from licensees who want to retain the proprietary status of their own contributions. For a more balanced approach, the CPL or OSL leave the term *derivative works* to be defined by the courts under copyright law.

- Does the license define *distribution*? The OSL goes farther than the other licenses described in this book by defining *external deployment*, so that the reciprocity provision applies regardless of how the derivative work is distributed. (See also the even more dramatic definition of *external deployment* in the Real Networks Public Source License published on the OSI website at *www.opensource.org*.)

- Consider the scope of any patent licenses you will grant. Many licenses have only implied patent grants; the scope of those licenses is unclear. As for licenses with explicit patent grants (i.e., the Mozilla, CPL, and OSL/AFL licenses), decide whether you wish to allow your patents to be used for creating derivative works; these licenses have subtly different patent grants.

- Are you prepared to grant a warranty of provenance (e.g., the OSL/AFL licenses, and similar "representations" in the MPL and CPL licenses), or do you prefer to disclaim all warranties? Remember that a disclaimer of warranties is not always effective in every jurisdiction, so if you intend to distribute open source software in some countries you may have to accept warranties regardless of what your license says.

- Also consider your disclaimer of liability. You should consult an attorney to determine your potential liability in all countries in which you intend to do business.

- Do you want a defense against patent infringement lawsuits? If so, should the defensive strategy terminate only patent licenses (i.e., the Mozilla and CPL licenses) or both copyright and patent licenses (i.e., the OSL/AFL licenses) for patent infringement claims? Is it sufficient to mandate an express condition that the software cannot be distributed if there is a patent infringement claim against the software (i.e., the GPL license)?

- Do you want your license to be interpreted under copyright law only (i.e., the GPL) or under both copyright and contract law (i.e., almost all other open source licenses)? If the latter, don't forget that it isn't only the license terms but the license formation issues—offer, acceptance, and consideration—that must be dealt with.

- Does the template license you use select a convenient and comfortable jurisdiction, venue, and governing law? If not, ask your attorney what the defaults are in your jurisdiction.

- Do you want an attorneys' fees provision in your license? Remember that, in most jurisdictions, such provisions apply equally to all parties to a contract. You are usually subject to paying attorneys' fees if you lose a lawsuit under a license with an attorneys' fees provision regardless of whether you're the licensor or the licensee.

These questions are intended merely to get you thinking about licensing alternatives. Your attorney should be consulted before you actually craft or select a license.

Contributions to Projects

Some open source projects seek copyright assignment from their contributors. This serves two purposes:

1. A project that owns copyrights has standing to enforce those copyrights in court without needing the contributor's participation or approval.

2. The project, and not the contributor, now has the right to make licensing decisions about the software.

Many authors of software refuse to assign their copyrights. The experience of musicians, photographers, writers, and artists in past generations warns us not to lightly give away that which we create. And the experiences of literally thousands of open source projects give us reason to believe that open source projects can thrive quite nicely with mere licenses from contributors rather than copyright assignments.

Contributors can license their contributions to projects, knowing and intending that the projects will combine contributions from many people, modify them in some coherent way, and then distribute a resulting derivative or collective work to the public. The terms of the contributor's license determine what the project can do with the software.

Software licensed to a project under an academic license can generally be used for any purpose whatsoever. It can be treated as a contribution to any open source project. For example, if software were blood, contributors under at least some academic licenses would be universal donors.

Through reciprocity, the GPL creates a commons of software similarly licensed under the GPL. That software can be combined and modified under the terms of the GPL by anyone and everyone, and so the license doesn't classify people as contributors or anything else. The GPL refers to all licensees as "you." Again, if software were blood, GPL-licensed software would all be of one blood type.

All GPL-licensed software is available for reuse in all projects using the GPL license. No separate contributor agreement is needed. However, some open source licenses deal more directly with the special characteristics of contributors. For example, the MPL distinguishes an "Initial Developer Grant" (MPL section 2.1) and a "Contributor Grant" (MPL section 2.2). These two sections of the license are almost identical, except that the Initial Developer contributes the Original Code and the Contributor contributes Modifications. Section 3.1 of the MPL makes the license reciprocal for Contributors.

Later in the license, the MPL, which was written by the Initial Developer and thus reflects that company's interests, sets more stringent conditions for Contributors. For example, the latter accept obligations regarding Intellectual Property Matters that don't necessarily apply to the Initial Developer (MPL section 3.4). The MPL also allows the Initial Developer—but not the Contributor—to designate alternative licenses under which portions of the Covered Code can be distributed. (MPL section 13 and Exhibit A.)

The CPL is more balanced than the MPL. The person or company who starts the software development process is merely the "initial Contributor" and everyone later is a "subsequent Contributor." The license grant extends to Recipient, who is defined as "anyone who receives the Program under this Agreement, including all Contributors." (CPL section 1.) Under the CPL, Contributors are simply those who distribute the Program. (CPL section 1.)

If a project distributes its software under the CPL, it can accept contributions licensed under the CPL. No separate contributor agreement is needed.

The OSL/AFL licenses apply the GPL's approach to contributor licenses—there simply are no distinctions drawn among types of licensors or licensees and no need for a separate contributor agreement. The Licensor is the owner of an Original Work, and the licensee is You. If software were blood, contributors under the AFL would be universal donors, and

OSL-licensed software (because of its reciprocity provision) would all be of one blood type.

License Compatibility for Collective Works

I finally explain Open Source Principle # 5, which states that "licensees are free to combine open source and other software."

The word *combine* in this present context means to "bring together or to join." This is a common activity in the software world. We do that when we load a variety of software onto our hard disk, perhaps from different vendors, to perform useful tasks. For example, business owners often combine an accounting package to collect and store financial data with a tax package that is used at year-end to calculate the government's due based upon those financial data. Office productivity suites may include separate programs for word processing, spreadsheets, and electronic mail. These software packages may actually communicate with each other so that data need be entered only once.

Distributors of open source software often aggregate separately developed contributions onto their distribution disks as a convenience for their customers. These contributions may have been designed originally by their authors to interact with other programs in the aggregation, and the original authors or downstream aggregators may even have tested them for compatibility. Or they may be compatible simply because the contributions were designed to meet industry standards.

Computer hardware and software vendors often build turnkey systems, combining operating systems, drivers, data bases, server software, utilities, applications, and other "glue" to create comprehensive customer solutions. Such combinations, under copyright law, are *collective works*.

It matters not whether some of the contributions to collective works are open source and some are proprietary. No open source license can prohibit a licensee from using an open source accounting package in a collective work with a proprietary tax package, or a GPL-licensed operating system with an Apache-licensed server and an MPL-licensed browser. Users are free to select open source software based upon technical criteria without restrictions as to the uses—or combinations of uses—to which that software can be put.

The contributions to a collective work always retain their original copyrights and licenses. (17 U.S.C. § 103[b].) If they are open source, contributions to a collective work can be removed and reused in other collective works, subject to the terms and conditions of their original open source licenses, even without the permission of the author of the first collective work.

On the other hand, a collective work as a whole is also an original work, subject to its own copyrights and its own license. Here's a simple example outside the software field: You may copy each of the public domain poems in an anthology of Chinese poetry, but you may not copy the anthology itself without permission of its author.

So it is with software. While you may remove and reuse the original open source contributions in a collective work, you may not copy or modify the collective work itself without the permission of its owner. For example, you may remove and redistribute Linux from the Red Hat or SuSE distribution disk, but you may not simply copy and distribute those companies' entire distribution disks—unless, as is usually the case for these open source distributors, the licenses for the distribution disks permit you to do so.

There is nothing in any open source license that would prevent someone from creating a non–open source collective work of open source software, trying thereby to collect royalties for copies of the collective work or to prevent people from making copies of the collective work as a whole. Of course, that can't affect the open source character of the individual contributions themselves; the collective work, however—reflecting the creative aspects of the aggregation process—may be copyrightable and restricted.

The aggregator remains responsible for honoring the terms and conditions of the licenses to the individual contributions he or she has collected together including, if necessary, publishing the source code of those contributions and making available copies of the relevant licenses.

License Compatibility for Derivative Works

If there is one issue that causes the most confusion and angst in the open source licensing community it is this: How do open source licenses interact with each other when derivative works are created from multiple contributions?

For example, a GPL-licensed contribution may be offered for an Apache-licensed derivative work. Or an OSL-licensed contribution may be offered for a GPL-licensed derivative work. What license terms apply to the resulting derivative work? Can the contribution even be accepted, consistent with the terms of both the contribution and derivative works licenses?

I discussed in the previous section the simpler problem of incorporating a contribution into a *collective work*; that is always allowed under an open source license because of Open Source Principle # 5. And I leave until Chapter 12 the complex issue of how you determine whether something is actually a derivative work. For present purposes, all I ask is whether the open source licenses are compatible for creating derivative works, whatever that technical term of art means.

License Compatibility for Contributions under Reciprocal Licenses

It is easy to understand what happens when you in-license a contribution under a reciprocal license. You can't use it for a derivative work unless both the contribution and the derivative work are licensed under the same reciprocal license. That is the very principle of reciprocity, as represented in the chart below:

		DERIVATIVE WORK			
		GPL	MPL	CPL	OSL
CONTRIBUTION	GPL	yes	no	no	no
	MPL	no	yes	no	no
	CPL	no	no	yes	no
	OSL	no	no	no	yes

This chart suggests that once you start a contribution under the GPL, MPL, CPL, or OSL, that same license is the only one that can be used for subsequent derivative works. In reality, however, the reciprocity provisions in open source licenses are much more subtle than that.

Some licenses, such as the MPL and CPL, complicate the analysis by defining an iterative process by which contributions become part of a package that grows over time. Those contributions are not necessarily separately licensed, and you have to analyze the license carefully to determine whether it is possible to reuse contributions to those packages in other separately developed derivative works other than under the terms of the MPL or CPL license.

For example, the MPL expects contributions (Modifications or files) to be governed by the terms of the MPL. (MPL section 3.1.) But the MPL then allows contributions to be reused as part of a "Larger Work." (MPL section 2.2[a].) The term *Larger Work* is defined in terms reminiscent of a collective work. (MPL section 1.7.) I read this to mean that MPL-licensed contributions can be used for differently licensed collective works but not for derivative works, which appears to be consistent with the chart above.

The MPL license provides another potential escape from the license incompatibility problem by allowing licensees to distribute derivative works under the licensee's choice of the MPL or an alternative license specified by the Initial Developer in its Exhibit A. The website of the Free Software Foundation (*www.fsf.org*) suggests that if the alternative license is the GPL, then that part of the program has a compatible license. Note, however, that this choice is only available to the Initial Developer, and that it applies only because the alternative license is the GPL, not the MPL. According to the Free Software Foundation, the MPL itself remains incompatible with the GPL.

The OSL states the reciprocity provision succinctly:

> *[Licensor grants You a license] to distribute copies of the Original Work and Derivative Works to the public, with the proviso that copies of Original Work or Derivative Works that You distribute shall be licensed under the Open Software License. (OSL section 1[c].)*

There are no exceptions. Derivative works may only be distributed under the OSL, regardless of the license on the contribution. Of course, the license on that contribution must authorize that:

> *Licensor warrants that [the contributions] are sublicensed to You under the terms of this License with the permission of the contributor(s) of those copyrights and patent rights. (OSL section 7.)*

Under the CPL, *Contributions* do not include "separate modules of software distributed in conjunction with the Program under their own license agreement." (CPL section 1.)

> *"Contribution" means: ... (b) in the case of each subsequent Contributor: i) changes to the Program, and ii) additions to the Program. (CPL section 1.)*

So under the CPL, a derivative work is created not by accepting a separate Contribution and combining it in some way with another work, but by making changes or additions to that other work. Furthermore, the CPL requires that a Contributor be the author and distributor of his or her own Contributions, meaning that the CPL does not allow sublicensed Contributions at all.

The GPL license is widely considered to be the most restrictive in this respect, because of the interaction of the following provisions:

> *You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License. (GPL section 2[b].)*
>
> *You may not impose any further restrictions on the recipients' exercise of the rights granted herein. (GPL section 6.)*

Derivative works of contributions submitted under the GPL *must* be distributed under the GPL, and you can't add any further restrictions. Once a chain of title is started for a contribution under the GPL, the GPL is the only license that can be used for subsequent derivative works.

License Compatibility for Contributions under Academic Licenses

What does it mean for the GPL to say that you can't add any further restrictions?

The BSD and MIT licenses are read to contain no conditions that could possibly interfere with any other license for derivative works. According to the Free Software Foundation, these licenses can be used for contributions to GPL-licensed derivative works, and I am aware of no open source project, under any license, that ever refuses BSD- and MIT-licensed contributions for creating derivative works. Such software can be used anywhere for any purpose.

The Free Software Foundation asserts that the Apache License, perhaps because of its provisions regarding the Apache trademark, is incompatible with the GPL. (But is a trademark exclusion, which states an essential rule under trademark law, an additional restriction that makes a license incompatible with the GPL?) Most contributors use the Apache license for contributions to Apache software and for nothing else. It is a shame that valuable Apache software is not being used for GPL-licensed derivative works simply because of the resistance to additional restrictions by the authors of the GPL.

The answer is not so simple for the Academic Free License. As I described in Chapter 9, the AFL contains several terms and conditions that are at least different from, if not contrary to, the provisions of other licenses. The AFL permits derivative works to be licensed under any license, but does that mean that AFL-licensed contributions can actually be so used without conflict with those other licenses?

Among the provisions of the AFL that are additional to those in the GPL are terms relating to the scope of the patent grant; the requirements regarding attribution rights; the warranty of provenance; provisions relating to jurisdiction, venue, governing law, and attorneys' fees; and, perhaps most contentious, the patent termination provision in AFL section 10.

Consider the effect on downstream licensees and sublicensees of a contribution originally licensed under the AFL with its patent termination provision. That provision protects the original Licensor, **A**, from patent infringement lawsuits by his or her licensees. Assume **A**'s contribution is used by another author, **D**, to create a derivative work. Obviously **D** is a licensee of **A,** and **D** cannot sue **A** for patent infringement without terminating the license. That much is straightforward under AFL section 10.

But the AFL is sublicensable, and so what happens when the derivative work is licensed by **D** to a downstream customer, **X**, under some different license that doesn't provide notice of the patent defense provision. That other license could be the GPL, one of the other open source licenses described in this book, or even a proprietary license. The AFL imposes no restrictions on that kind of downstream sublicensing. **A**'s contribution is effectively sublicensed to **X**.

Can **X** sue the author of **A** for patent infringement without risking termination of his license for **D**? Does **X** even have any way to know of the terms of **A**'s license? Does section 10 of the AFL extend through sublicensing to protect the author of **A** even against patent infringement lawsuits by downstream sublicensees like **X**? Similar questions could be framed about other potentially uncomfortable terms from **A**'s license, such as the AFL's attorneys' fees provision or the scope of its patent grant. Do such terms bind—via sublicensing—the recipients of derivative works of AFL-licensed contributions?

I find it hard to believe that any court would bind any downstream sublicensee of an open source contribution to any terms and conditions of a license of which he was not informed and didn't manifestly accept. That is certainly a basic tenet of contract law and a fair result in the context of mass-marketed open source software offered for free over the Internet. So to the extent that an AFL-licensed component was sublicensed by **D** as part of a derivative work, customer **X** at the end of the chain cannot be bound to the AFL but only to the license with **D** that he or she accepted.

This situation is not unfair to **A**. Remember that **A** could have avoided this result by distributing his or her contribution under a license that forbids sublicensing. Instead, **A** intended to contribute software under a license that was completely permissive about derivative works. **A**'s software can even be used in proprietary derivative works. License terms do not pass through via sublicensing unless **A** insists upon it in the software license, and the AFL does no such thing.

So it is unclear to me how an academic license such as the AFL can be incompatible with any other open source licenses. The AFL doesn't impose any conditions except upon the licensee of that software, and that licensee is permitted to sublicense the contribution under any license whatsoever.

Of course, these notions of fairness and the requirement that a licensee be informed of conditions to which he or she is bound apply only under contract law, not for a bare license under copyright law. I don't know how a court would decide such sublicensing issues for bare licenses.

The MPL license deals with license compatibility for derivative works by requiring a specific Contributor Grant. As long as the Contributor submits his or her Modification under the terms of that Contributor Grant, the MPL doesn't care about other licenses. It is up to the Contributor to ensure that whatever he or she contributes is Licensable by Contributor. (MPL section 2.2.)

> *"Licenseable" means having the right to grant, to the maximum extent possible, whether at the time of the initial grant or subsequently acquired, any and all of the rights conveyed herein. (MPL section 1.8.1.)*

The CPL permits only Contributions that are original to the Contributor. Sublicensed Contributions aren't accepted. (CPL section 1.)

The OSL does not expressly prohibit the imposition of "further restrictions," nor does it deal separately with contributors. But it does contain the following warranty of provenance that has the effect of promising compatibility of licensing for all contributions incorporated into the derivative work:

> *Licensor warrants that the copyright in and to the Original Work and to the patent rights granted herein by Licensor are owned by the Licensor or are sublicensed to You under the terms of this License with the permission of the contributor(s) of those copyrights and patent rights. (OSL/AFL section 7.)*

A licensor promises that he or she has permission (i.e., licenses) to distribute those contributions in an Original Work under the OSL. The OSL handles the license incompatibility problem by placing on the creator of a derivative work an obligation to ensure that whatever contributions he or she accepts are authorized for inclusion in that derivative work to be licensed under the OSL.

		DERIVATIVE WORK				
		GPL	**MPL**	**CPL**	**OSL**	**Academic**
CONTRIBUTION	**BSD**	yes	no[1]	no[2]	yes	yes[3]
	MIT	yes	no[1]	no[2]	yes	yes[3]
	Apache	yes[4]	no[1]	no[2]	yes	yes[3]
	AFL	yes[4]	no[1]	no[2]	yes	yes[3]

[1] MPL section 2.2 is a Contributor Grant that expresses the terms under which contributions can be accepted for MPL-licensed derivative works.

[2] CPL section 1 defines Contributor and Contribution. "Separate modules of software" are not Contributions.

[3] The Apache Software Foundation now requires a Contributor Agreement. (See *www.apache.org*.) Other projects using academic licenses may also require contributor agreements or specific contribution licenses.

[4] The Free Software Foundation says the Apache and AFL licenses are not compatible with the GPL. (See *www.fsf.org*.) I disagree with them, and so I wrote YES in these boxes.

The interrelationships between the contributions and derivative works are summarized in the preceding chart. But there are so many caveats in the footnotes that this chart should *not* be used in a mechanical fashion. Review the contributor and derivative works licenses carefully to ensure that the terms and conditions of both licenses are honored.

In summary, the creation of derivative works from contributions under academic licenses depends more on the license of the derivative work than on the terms of the academic license. Some licenses won't permit the incorporation of works licensed under an academic license regardless of what the academic license itself permits.

Relicensing

For some of us, the problem of combining software under different licenses into derivative works is a frustration. License incompatibilities prevent software from being freely used and combined. And with the proliferation of open source licenses, the problem is getting worse, not better.

But copyright and contract law is unambiguous: Open source distributors cannot simply relicense other people's copyrighted software unless they have permission to do so.

One way out is to convince contributors to make their works available under a different license. This might be possible for small projects where there are few contributors who need to agree on a licensing strategy. But convincing everyone in a large project to reconsider their licensing is very difficult.

Are projects, by virtue of the licenses under which they received contributions, prevented from relicensing their derivative works to replace licenses they no longer want in favor of different licenses? Can relicensing be done by projects to make their works compatible with other contributor licenses?

There is a legal answer and a political answer and, for this particular question, the political answer is far more significant. Open source must be a collaborative process. Any licensing change that is made by fiat is likely to result in a fracture of the community. A project may be left without some of its key contributors. Customers will face diverging product development strategies by different groups of developers, each competing for attention and support. Entire product lines may die.

Among the difficult options for software projects that won't relicense by consensus to accommodate contributions for derivative works is to avoid making derivative works. This is essentially what the Free Software Foundation suggests in order to live with Apache despite its incompatible license:

> *We urge you not to use the Apache licenses for software you write. However, there is no reason to avoid running programs that have been released under this license, such as Apache. (http://www.fsf.org/)*

By merely aggregating software from different sources and treating such software as black boxes, one can technically avoid—sometimes with much clumsiness—creating derivative works. One can benefit from the software without actually having it available for internal modification and improvement.

This is not so different from what happens with proprietary software products. At some point, customers may demand different licensing terms than the licensor will provide. The choice is obvious: Live within the available license, or find different software.

Sometimes, where derivative works are prohibited, people write special plug-ins, drivers, or other complex workarounds to add functionality to programs they can't freely modify. When software vendors are particularly uncooperative with their licensing terms, creative people simply start from scratch and write the software anew under more favorable licenses.

License incompatibilities are inconveniences rather than barriers. Ultimately, one can get around almost all licensing restrictions to almost all intellectual property by being sufficiently creative and inventive.