Sun Community Source License Principles

by Richard P. Gabriel and William N. Joy

Executive Summary

Community Source creates a community of widely available software source code just as does the Open Source model, but with two significant differences requested by our licensees, as follows:

- compatibility among deployed versions of the software is required and enforced through testing
- proprietary modifications and extensions including performance improvements are allowed

These important differences and other details make Community Source a powerful combination of the best of the ***proprietary licensing*** and the more contemporary ***open source technology licensing*** models.

❧ ❧ ❧

In the contemporary world of Internet business, the traditional principles of overly protective software ownership don't make as much sense as they used to. Today it is difficult for a single company to house all the expertise it needs to succeed. Especially when a company wishes to build infrastructure on which other businesses depend, it cannot presume to know how best to do that right out of the chute. The theory of separation of concerns and modularity work well once the dividing lines are known, but finding those lines is a collaborative effort, requiring cooperation between companies.

As business practice has matured in the Internet world, it has become clear that there is good reason to expect that companies will be able to know when to cooperate, and so it makes sense to look at how to license software to take advantage of this emerging understanding.

The Sun Community Source License (SCSL) is designed to balance the needs of organizations needing to innovate rapidly in order to grow with the needs of those organizations to leverage a community's expertise while maintaining proprietary advantages.

Historical Perspective

In the recent past there have been two alternative approaches to software licensing: traditional ***proprietary licensing*** and ***Open Source licensing***. Open Source licensing is relatively new, being preceded by shareware and free software.

Proprietary Licensing

Proprietary licensing recognizes primarily binary use licenses which restrict software to execute-only formats, and only those who have bought a license are entitled to use the software. In some cases, the source to the code is available at extra charge and only for limited purposes. In this model, the value of the software is assumed to be contained entirely within that software, and as such, it is to be protected as dearly as any other company asset. Such a model has several advantages, as follows:

- It provides protection for intellectual property.
- It guarantees ***structured innovation***, which is innovation that is planned within a single responsible organization.

- It is clear who owns what.

On the other hand, proprietary licensing has a number of disadvantages revealed by the pace of Internet business, as follows:

- Innovation and releases are scheduled, and the schedule may be ill timed for companies that depend on the software. Further, even when a schedule is acceptable for dependent companies, schedules can slip.
- Binary-only products are classical black boxes, and the problem with a black box in an emerging sector is that the wrong things might be in the black box: code with a wrong performance profile, improper resource utilization, or over- or under-generalization.
- The quality of the software depends entirely on the owner organization; this can be a problem if the owner organization does not place the same value on quality as the using organization, the owner organization has a different perspective on quality, or if quality resources are not allocated in ways that the using organization needs.

In short, the benefits naturally fall out of the observation that proprietary software is completely black box, and that when the boundaries of the black box are optimally placed, the benefits are clear and effective. But if those boundaries are ill-placed or releases are ill-timed, the result is frustration and lost opportunities.

Open Source Licensing

Open Source licensing recognizes that different organizations have different concerns, and therefore the source code is freely available for any party to do what it will. Open Source licensing grew out of the tradition of shareware and free software, which were movements that embodied political beliefs about what can and cannot be owned.

Open Source licensing, on the other hand, does not embody beliefs about ownership aside from recognizing the fact that some organizations wish to own software and others don't. However, Open Source licensing does recognize that certain common elements—for example, infrastructure—are important for a number of parties and that those elements should be freely and openly available.

With Open Source licensing, the source code is available for any use, but there are incentives to "give back" to the Open Source community those improvements that are for the common good. Most importantly, the Open Source approach recognizes that the primary value of a piece of software is the expertise represented by the people who developed it. Thus, even when source is out in the open, the value still remains with those who can expertly manipulate it.

Such a model has several advantages, as follows:

- The code is open with published and, often, specified interfaces.
- There are more developers looking and working on the common source code, so there is higher quality and more-rapid innovation.
- There is no central owning organization that sets schedules and priorities that might conflict with a using organization's schedules and priorities.
- There is a self-organizing effect in which the boundaries between proprietary concerns and community concerns are adaptively set.
- A participating organization can reap the benefits of expertise not in its employ.

There are disadvantages as well, as follows:

- There is no clear control over compatibility issues and there may, therefore, be fragmentation.

- There may be no responsible organization. Bugs introduced by another organization may be too difficult for a using organization to fix and of too low priority for the author to fix in a timely manner.
- Progress can be chaotic and undirected.
- There are limited financial incentives for improvements and innovations, leading commercial developers to use the proprietary model.

Community Source License

The Community Source licensing model takes the advantages from each of the proprietary and Open Source models and eliminates the disadvantages.

In Community Source licensing there is a community of common interests centered around an infrastructure which is provided by a particular organization, called the ***developing organization***. A group of other organizations may join the community, and generally those organizations will have an interest in building businesses around the infrastructure.

For example, Jini technology licensing is based on a community of companies who wish to build and sell Jini devices. Sun Microsystems is the developing organization which invented, designed, and built the initial Jini infrastructure. This infrastructure is network based and provides a mechanism for devices and software services to gather into a spontaneous network of capabilities.

The community comprises those who have agreed to the license, and within this community there is a mostly Open Source model of interaction. However, because the members of the community are bound by a common license, intellectual property is maintained and there is no requirement to share openly everything developed for the infrastructure. Moreover, the community comprises only those who have agreed to the license, not the general public.

Three levels of participation in the license can be supported, as follows:

- Research Use, which is not only for real research—for example, to improve the infrastructure—but for evaluating the technology and prototyping potential products.
- Internal Deployment, which is both for (very) limited distribution and for testing before launching a product.
- Commercial Use, which is for selling products.

Principles

The following summarizes the principles behind the design of the Sun Community Source License.

Immediate, Open Access

In order to encourage members to join the community defined by the license, there is an easy and risk-free way for a company to get access to the infrastructure source code. Any company or developer can become a Research Use licensee with a simple click-through license which grants broad experimentation and evaluation rights. The licensee may use the source code and its specifications in any way whatsoever short of deployment.

With a Research Use license, a developer may take any approach to using the source code in order to determine whether the technology can be of use; in fact, the developer can do all the work leading up to deployment with such a license.

The license and source code are available through the Net, and so the transition from interest to participation is immediate. Access is clearly open.

Increased Innovation

In order to increase the rate of innovation, improvements and additions to the original source code may be incorporated into the source code regardless of where they come from. There are two basic types of source code improvements that can be made: improvements to the core technology and improvements or additions to surrounding technology. With Jini technology, for instance, there is the basic infrastructure and there are services. Each sort of service—storage services, for example—will have common elements which can be improved by community efforts. In this case, acceptable modifications and additions are determined by the subcommunity of organizations who are working in that area using an open process. In such cases it is expected that change can be very rapid, and indeed, must be in order to gain widespread agreement and deployment.

With infrastructure modifications and additions, change is expected to be less rapid because the base of organizations that depend on that infrastructure is large. However, because developers at organizations with different requirements are working with the core technology, over time it is expected that situations will arise that the original developers did not anticipate. These outside developers are able to evolve the technology for their products and markets more rapidly than the original developers would ordinarily be able to do using a traditional proprietary license.

In short, there is a spectrum from the innermost portions of the infrastructure where change should be slow and deliberate to the outermost portions—which might represent applications—where change should be as rapid as the market requires.

Increased Work Force

In order to increase the effective work force, participation is not limited: Any organization can participate in the Research Use License without paying or negotiation.

Increased Quality

Simply stated, with more eyes looking at the code, there is more chance that errors will be caught and repaired, particularly when that code is used in situations and contexts not anticipated by the original developers and researchers. It might be thought that with an Open-source type of model there would be increased likelihood of errors in code, but the Open Source experience is largely the opposite—that because the source is published, developers are generally very careful while developing code in order to avoid public embarrassment.

Moreover, the test code is also covered by the license, and the benefits of a larger community working accrue to the test suite as well as to the target technology.

Faster Commercialization

The Internal Deployment License makes the transition to commercial use easy. Once the technology has been evaluated and adapted through the Research License it can be deployed internally by conforming to the related test suites and specifications. In some cases we expect that Internal Deployment will be allowed without a fee, as it is for the core of the Jini technology. The Internal Deployment License is made available with the Research Use License, which means it is also immediate and easy.

The Commercial License is separate from the Research and Internal Deployment Licenses. It is tailored to provide a dependable platform for all third parties and to provide revenues to the developing organization, which can be used to fund both support for the entire community of licensees and also further development.

Access to Students

Researchers, teachers, and their students are particularly attuned to using technology at its limits, and therefore such people are especially encouraged to participate by special conditions of the SCSL. It is easy to use the source code in laboratories and in the classroom.

Protection for Intellectual Property

Licensees rightly enter into the community expecting that their intellectual property rights will be respected. Therefore it is not required that a participant give up intellectual property rights when joining the community. In order that the community remain open, however, the SCSL does require that any programming interfaces which extend the platform or infrastructure technology be open and specified, even while implementations and intellectual property embedded in the implementations may be held proprietary.

The Best of Both Worlds

The Sun Community Source License blends the best aspects of the proprietary and Open Source license models. The SCSL was designed to support a ***developing organization*** and a surrounding ***community of participants***. The developing organization is generally building an infrastructure that will spawn a number of new marketplaces or business opportunities for the community of participants. The developing organization might itself act also as an ordinary participant in the community if, for example, that organization desires to take advantage of a new marketplace or business opportunity. Nevertheless, the two roles are separated in the SCSL. The developing organization deserves to reap some benefits for developing the infrastructure, and the SCSL recognizes that.

The following lists the benefits of the SCSL that are shared with or derived from the proprietary model:

- It provides protection for intellectual property.

  While error corrections to the licensed technology must be given back to the community, other modifications can remain proprietary at the discretion of the participating organization that created them.

- It guarantees structured innovation within a single responsible organization.

  The developing organization is responsible for the original code base and the community is responsible for the contributed portions. Ultimately, the developing organization is responsible for the entire source code base including moving it forward.

- It is clear who owns what.

  The infrastructure part of the original code and upgrades to it are owned by the developing organization and shared with the community. Error corrections are required to be given back to the community by all licensees. Community members may contribute ***shared modifications*** to the community, granting rights to the community to use these modifications, including intellectual property, specifications, and test suites.

  Licensees may make modifications, such as performance enhancements or platform adaptations which are compatible with the licensed technology, or make extensions—these modifications and extensions may be kept proprietary, though extensions are required to have open interfaces.

- There is clear control over compatibility.

  For the infrastructure part of the original code and upgrades to it, the developing organization provides specifications and test suites. These test suites must be passed by all internally distributed code. Commercial distributions have an additional requirement to use relatively recent upgraded code and pass these test suites, so that the platform can evolve.

  For extensions, the community member making the extension is required to supply specifications and, ultimately, test suites.

In some cases neither a test suite nor a good specification may be enough to guarantee that a modification, extension, or bug fix is compatible. For some purposes a ***reference implementation*** can provide not only a base on which to build a commercial version of a specified piece of functionality but a guide to ensure correctness. Although the SCSL does not mention reference implementations per se, reference implementations as part of a process of defining and accepting specifications for interfaces, for example, are compatible with the SCSL principles and philosophy.

Benefits SCSL shares with or are dervied from the Open Source model are as follows:

- The platform is open with published and specified interfaces.

  The SCSL recognizes the strong community interest in an open platform. A primary mechanism for extension within the community is publishing the specifications for new and often layered interfaces. These interfaces may be best—or in rare cases, only—implemented by using proprietary techniques and technology; this is permitted, but under the SCSL, the programming interfaces themselves and related specifications and test suites must be open.

- There are more developers looking and working on the common source code, so there is higher quality and more-rapid innovation.

  The community pulls organizations and developers into a circle of shared concerns, and this community can effectively self-organize with only a little assistance from the developing organization.

- There is no central owning organization that sets schedules and priorities that might conflict with a using organization's schedules and priorities.

  Though there is a schedule for the developing organization's structured innovation, the SCSL provides every participant with all the freedom and authority to move forward independently.

- There is a self-organizing effect in which the boundaries between proprietary concerns and community concerns are adaptively set.

  There is a force tending to keep the infrastructure stable because all the participants depend on it, but innovation can take place any place in the technology base. The binding requirements of the SCSL such as openness and compatibility also tend to keep the infrastructure stable by preventing predatory modifications and extensions.

  There is also a spectrum of concerns from the developing organization's point of view that ranges from extensions of high importance to those of less importance; by initiating work for those high importance items itself, the developing organization can exert beneficial stability on them.

- A participating organization can reap the benefits of expertise not in its employ.

  There is no limit on who can participate.

Additional benefits to the SCSL are as follows:

- Proprietary modifications including performance improvements are allowed.

  Unlike the proprietary licensing model, all community members can make such changes to the original source base (or upgrades to it) because they have access to it. Unlike the Open Source model, such changes are not required to be given back to the community, though it is encouraged. (Note however, that error corrections to the original source code base must always be returned to the community.)

- Innovation and releases are not subject to a centrally planned schedule.

Each participating organization may innovate or release at any time with any combination of private and community sources. In order to ensure that other independent releases are compatible, a conformance suite must be passed before release, and published specifications must be respected. For simplicity, any necessary testing and certification can be performed by the releasing participant.

- The quality of the software can be improved throughout the community.

  Each participant can determine appropriate quality levels and work toward them independent of any other participant or in conjunction with others.

- Progress is always made on the original source base, and progress on the peripheries depends on participant needs.

  The original source base is fully the responsibility of the developing organization which can move that base forward; the community may participate in those activities, accelerating them as necessary. Because commercial users of the technology are required to update to newer base technology in new products, after a reasonable delay, the platform can evolve.

  As interest builds in creating related technologies, those parts can move ahead at their own pace.

- There is a place for both incremental improvment and reward for invention.

  Not only does the developing organization have sufficient motivation to invent and innovate but for participating community members there are incentives to invent in the marketplace and protections for those inventions within the community. That is, by creating a more protected community than a completely public one, there can be protections tuned to encourage and safeguard innovation and invention within that community.

Effects

In the last few years there have been remarkable changes in the computer industry largely spurred by the Internet, the Web, and by alternative forms of business. For the first time we have seen business strategies based on giving away products as rapidly as possible, we've seen Open Source gain a large measure of success, and we've seen collaboration where common interests play a strong role.

There are many benefits to the sort of license the Community Source License represents. The following sections talk about them a bit.

Increased Commitment

One of the difficulties of trying to get a new infrastructure technology adopted is the sense that a proprietary standard might not make it or that a competitive situation will make choosing it difficult or impossible. Regardless of who the sponsoring company is, there is a risk associated with not knowing everything.

Because participants in the Community Source License have access to the source code and can reasonably expect that their innovations will appear in that source, those participants can see everything there is to see about the technology. By being part of a community of partners, the risk of putting all one's eggs in a single company's basket is reduced. The SCSL creates a context in which commitment is less risky, and the more participants there are, the less risky it is.

Closer Ties Between Development Groups

Developers have always gotten together at conferences and coffee houses, and when they do they are very good at two things: exchanging useful information in the form of techniques and know-how,

and not revealing secrets. When developers get together within the community created by the SCSL, they are able to speak freely about particular issues represented by the shared source code. Thus, beneficial information exchange can take place more rapidly, and each participating organization benefits—because writing software is hard.

Rapid Spread of Influence

Because it is so easy for a participant to join the community and because there is a shared commitment to the technology once participants are fully engaged, the underlying infrastructure is spread more rapidly. And because there are more development groups pushing up both the innovation and quality, the new marketplace based on the infrastructure is created more rapidly than it would any other way.

Reduced Risk for Adopters

Frequently, black-box technology products pose too great a risk for many organizations because those organizations end up depending on another company for the development, maintenance, and evolution of parts of their own products. Most importantly, the priorities the owner company would assign to work related to the technology are not necessarily priorities a customer would choose. With SCSL, the participants share common source code and can set their own priorities, working as much or as little on the code as they wish.

Those organizations that require a more stable infrastructure can take it from a platform provider as usual just as some organizations prefer to buy from Red Hat rather than getting Linux from an ftp site.

The Future

In writers' workshops, in design charrettes, in code reviews and walkthroughs, and in artists' studios there is one common theme: By working within a community, a better result is achieved than working alone—even virtuosos learned by criticism and sharing. And if you talk to almost any artist or software developer, you will hear that what they are doing is a "work in progress."

In defining the Sun Community Source License, Sun has tried very hard to get it right, but maybe it isn't perfect. Naturally, we believe so much in our open process that the license itself is subject to that same process, and as participants need changes in the license, we will make them.

Right now, in the context of an industry rapidly changing, it's our best attempt—it's a work in progress.