BUSINESS ISSUES IN FREE SOFTWARE LICENSING

Donald K. Rosenberg, *Stromian Technologies*

I. BACKGROUND: FREE SOFTWARE MOVEMENT AND SOFTWARE VENDORS

There are some odd ideas circulating about Linux and the Free Software or Open Source movements. We can call these ideas primitive because they are simplistic and not well thought-out, and because they go back to the *reductio ad absurdum* of the primitive peoples who believed (and still believe, we hear) in what anthropologists like to talk about as the "cargo cults."

According to the anthropologists, these movements began among South Seas peoples in the 19th century, when they awaited the arrival of large ships which would restore to them all the wonderful goods their peoples had owned once, long ago. After World War II these cults took the form of waiting for aircraft to descend from the skies with their abundant cargoes. We are told that the believers even constructed runways with mock aircraft on them, hoping to attract the passing air traffic.

We all smile—how much more we know than they—but today there is a firm body of thought on the one hand that eventually all software in the future will be produced, shared, and enjoyed on the Bazaar model: freely developed and given away by loosely-organized programmers around the world, and superior in quality and design to the commercial products of today. Given the behavior of much modern commercial software, one can understand why the believers hope so fervently for the millennium.

But commercial vendors are just as likely to make the same mistake: we see articles and columnists hyping the idea that if a software firm can just take the leap of faith into arms of Open Source, they will attract legions of the world's smartest programmers, working ceaselessly and without compensation to improve the code the vendor has thrown among them. It does not help that any announcement that a company is releasing source code is regarded by the business community as a desperate act of last resort.

What should be the approach of a commercial software vendor to the Open Source space? And what do they really want, anyway?

II. WHAT ARE SOFTWARE VENDORS TRYING TO ACCOMPLISH?

Software vendors want to a) protect their financial investment in their code, product, and channels of distribution and b) to recover that investment, along with a profit. Somehow they want to make sure their work is not appropriated, and that there will be revenues which will keep the doors open and the families fed.

These two themes—protection of property and recovery of investment—will dominate the rest of the talk.

III. HOW CAN THEY DO IT?

A. Current Open Source Models

Protecting the property and tapping the correct place in the distribution stream for revenue are the chief purposes of commercial software licensing. There is a great variety of licenses available—the key thing to remember is that they should reflect the business goals of the vendor.

GNU General Public License

The goals of the Free Software Foundation are to keep its GNU software (and any other software using the GNU GPL) completely free and Open Source. Nevertheless, many vendors can make money from it by providing related services. Two highly-successful examples are O'Reilly & Associates, who publish books about Open Source products, and Red Hat Software, a branding company described by its Chairman, Bob Young, as "a company that gives away its software and sells its sales promotion items." Red Hat distributes the Linux OS for money on CD (as well as for free from its Web site), and like numerous other companies, sells software support.

The GPL is a good thing for an operating system: it is something that all users want kept free and open; all developers want equal access to the system so that none can take advantage of operating system secrets. Most tool and application vendors, however, will not want to go into the Open Source arena until they understand how it works, and how they are going to structure their company, products, and services to make money in this arena.

There are different ways to go about this; to pick just a couple of examples of products that make their source code available for free, let's look at Scriptics and Aladdin. Both succeed by segmenting their product line into free product and revenue product, on the basis of licensing.

Scriptics

The popular scripting language and toolkit, Tcl/Tk, claims a million users. Scriptics is a new firm founded by the developer to commercialize Tcl/Tk. While keeping the core material and its improvements free, the firm will develop niche applications for money. Because the user has the right to modify and distribute the source code, John Osterhout recognizes that he has to keep the free side of his business happy in order for Scriptics to remain the center of Tcl/Tk development. He intends to do this by keeping the scripting language and toolkit good enough so that the free side attracts new users. In turn the Scriptics Web site aims to be the principal online resource for Tcl/Tk, attracting prospects and converting them to users. Profits from the commercial side of the operation will pay for the in-house developers working on Open Source Tcl/Tk.

Thus Scriptics divides the free/revenue sides of the business by focussing one on core technology, and the other on niche adaptations of or extensions to that core.

Aladdin

Aladdin's Ghostscript splits itself into different free/revenue versions using different licenses: a separate enterprise called Artifex distributes a commercial version called Aladdin Ghostscript, and there are free versions under the Aladdin Free Public License and GNU Ghostscript. The Aladdin Free Public License resembles the GPL and has additional restrictions. You can't accept money for the free program except for cost of disks and copying, you can't put the free version on a disk with any paid-for software; the bundling restriction helps kill commercial distribution of the free product. On the other hand, the licensed commercial user can use Ghostscript in his application and also get interim updates to the code; free users wait for the annual update.

Thus Aladdin divides the business into free/revenue by segmenting the technology into core technology vs. paid latest updates to that core.

But these examples do not mean that any vendor can release the source code for a product, and expect the community to enthusiastically pick it up and improve it. The vendor needs to provide a good infrastructure for maintaining the source tree, including bug lists and version control, and in all the enterprises named above—Red Hat, Scriptics, and Aladdin—there are paid staffs of programmers who maintain and improve the source code, and then make it available (at some point) for free to the community. All of this effort is not merely a matter of being seen to cooperate with the Open Source community by making contributions; it is all part of maintaining a leadership position as the authoritative source of the free product.

Different licenses are used to maintain different degrees of control in upholding this leadership. Red Hat and Scriptics ride bareback: Red Hat faces competition from other Linux distributions, and by offering a better version of Tcl/Tk, it is theoretically possible for another party to make Scriptics a secondary player in the free version of Tcl/Tk. Much more restrictive are the recent licenses such as the Sun Community Source License, which makes it clear that although the source is open to examination, modifications will be tightly controlled by Sun through a testing and revenue-license program, and that only Sun may maintain a source tree for the code. It will be interesting to see what success these restrictive licenses will enjoy.

Restrictive as the Sun Community Source License is, it is at least fairly clear when it comes to the user. A more difficult problem has been introduced into the Linux community by the free or public versions of the license for the Troll Tech Qt library.

Q Public License

The library is a toolkit: the user is free to use and distribute it and his derivative application--unless it's a commercial application. Commercial distribution is effectively stopped by requiring the derivative application to give away its source code and permit further distribution and modification. Commercial users must buy the Qt toolkit under the Professional Edition License; the developer pays a round sum for the Qt toolkit and the right to distribute runtimes.

The chief objection to the old Qt Free Edition License was that it did not permit the Qt toolkit itself to be modified in any way when it was distributed. The license also contained incorrect and confusing language about use of the GPL or a BSD-type license as alternatives. These are the perils of writing your own license and not getting it right. The new QPL, however, has cleared away the confusion and now also permits the distribution of separate patches to the Qt source code, so the Open Source Initiative (OSI) (www.opensource.org) says the license now meets the Open Source Definition (OSD).

There are still plenty of voices objecting, however, to Qt's licensing, even under the new QPL, and the voices are raised because Qt is the technology underlying

KDE, the highly successful free Linux desktop. Note that a developer can never use the Q Public License on anything but some sort of UNIX platform (basically Linux), and that Windows (and the Mac) are reserved for the Professional Edition.

The motives are clear: Troll Tech wants to control the toolkit, and to prevent forking; therefore, no modifications are permitted. The firm makes the product free on Linux in hope of collecting improvements from users, and wants to reserve the Windows and Macintosh platforms for their revenue product. Troll Tech wants to use the Q Public License to promote their technology, spread its use and familiarity, and lure people (some will say trick them) to the Professional Edition.

B. Licensing Dependencies

In some respects the business model for the Qt toolkit is not unlike other free/commercial segmentation. If you want to distribute for money, you pay Troll Tech money. But the stinger is that persons developing software on the free and highly popular KDE desktop suddenly find that they owe money to a third party, Troll Tech, for use of the underlying QT toolkit. As long as the application was free, there was no problem; as soon as the developer wanted make money (or even just to collect a little revenue to cover his costs), the licensing terms undergo a change of state that is working to undermine the formerly unified Linux desktop.

This is an issue of license dependencies; it can be a problem as bad as software dependencies.

In the case of Qt, the solution to the dilemma may not include the survival of Troll Tech. Although some people think Qt is such great technology that Troll Tech will be able to get what they want, others are so horrified by this problem that we have forking in what had seemed to be a common Linux desktop. Debian dropped Qt and KDE from its distribution, posting an explanation on the site; Eric Troan explained on the Red Hat site that Red Hat could not put Qt and KDE software on the basic development CD if its licensing terms were so different from the other development software there.

Finally, resentment about Qt's licensing has caused movements to spring up to clone Qt. Harmony, a project to clone a Free Qt, is still active, I've been told, and the GNOME movement has sprung up to put out a rival toolkit distributed under the GPL and LGPL, just like Linux. There are desktop projects based on GNOME, and Red Hat is working on one of them. It will take the Linux community a while to get over this split, which has its origins in licensing dependency.

At the end of this paper is a diagram of a simple scheme of purposes and dependencies:

Base Layer – Operating System – GPL

Operating systems stand to benefit the most from the GPL because they are the broadest-base software; the users of an operating system will always outnumber the users of any particular application on that system. There is more choice in applications than in operating systems. The GPL may do its best work at this level, forcing a standardization of all licenses, and aggressively keeping it open and free of all closed material. Openness is highly important for operating systems; applications can get away with more-limiting licenses because their usage is more limited.

In this Base Layer of the OS, vendors can earn money on source code distribution, and they can earn money on binary distribution (so long as source code goes out with it). Linux distributions using the GPL are more or less successful businesses

Second Layer, Part A: Toolkits

At the next level, software may be either free or commercial, but it is essential that there be a firewall here (the vertical red division); the same toolkit should not be capable of changing state. The firewall represents not a separation of products for free/commercial, but clear licenses for those products. Tcl/Tk, for instance, lets you use it for commercial products or for free products.

Second Layer, Part B: Extensions and Libraries for Toolkits

At this next higher level, extensions and libraries need to adopt the same licensing (either free or commercial) as the toolkits they serve. Otherwise we're back in the "checkerboard" or "change-of-state" license. Developers must watch licensing dependency as closely as software dependencies.

Third Layer: Tools and Applications

Of the applications which are not tools, we can expect a larger proportion of these to be binary or proprietary. The proprietary applications can build upon both proprietary and free foundations, provided they respect the licensing of the layers upon which they are built.

Tools, however, need to follow the same choice as toolkits, being either on the Free side or the Proprietary side, so that their products likewise have unambiguous licenses.

At this third level, vendors have a strong desire for proprietary code to protect their development investment, and distribution in binaries is common for many products. The LGPL is often not enough to allow for use of proprietary code, and so BSD-style licenses fit here, as do products like the Apache-based Stronghold. The Aladdin Free Public License operates at this level in the Free category, while the Perl Artistic License, which is useable for closed, embedded commercial work fits into the Free and Proprietary category (we might ask whether this license would work as well for an application as it does for a language/script).

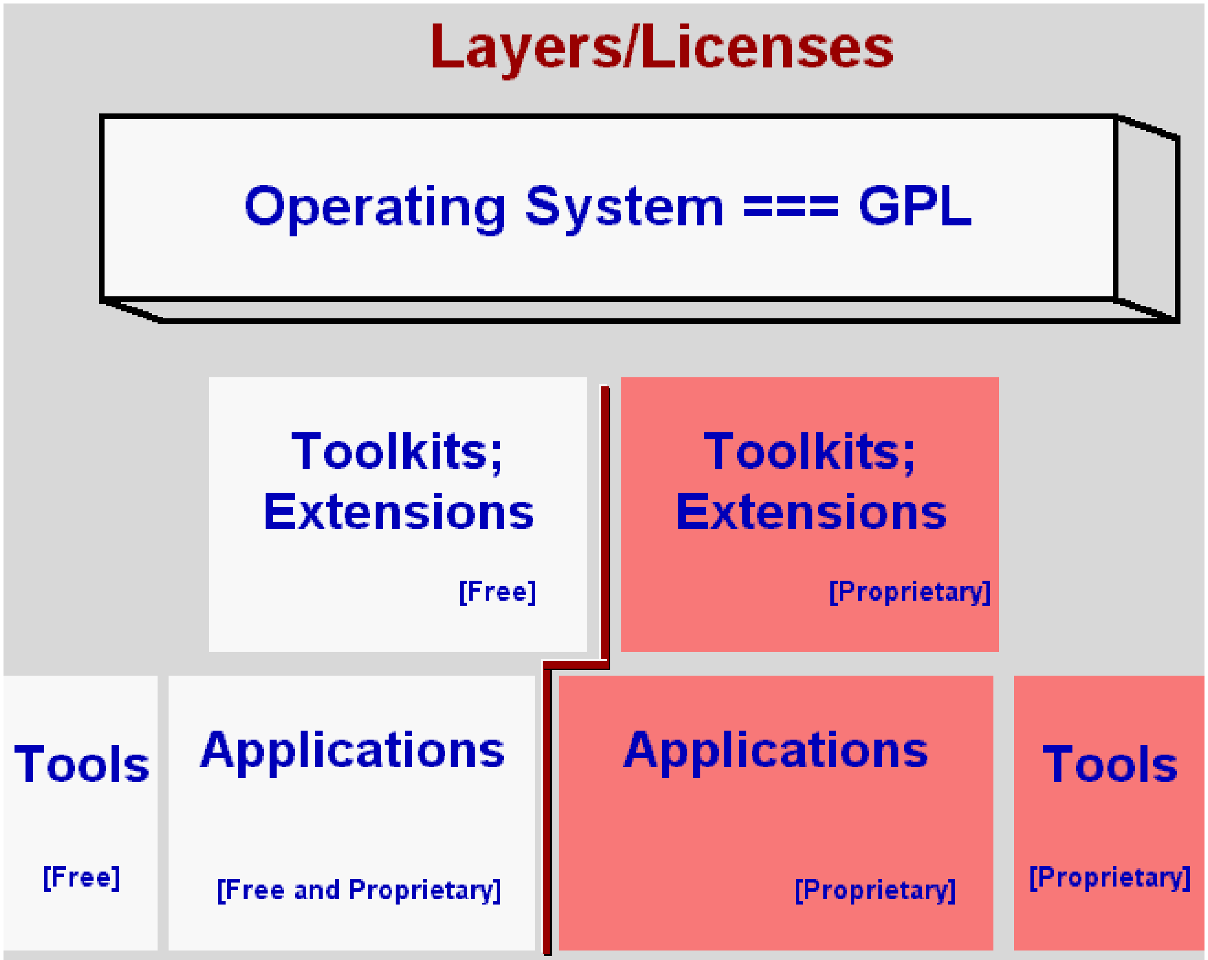
C. Further Innovation and Cooperation Between Commercial and Open Source Software

So far we've talked about some of the common ways a software vendor can ally with the Open Source movement (such as distribution and support), and we've seen that a license must be carefully written to achieve the business goals of the licensor. We've also seen that it is not a matter of tossing the software out there and expecting brownies to work on it during the night—a software vendor must provide the infrastructure and encouragement in order to form a coding community around the software.

This brings us to the question of a strategy for use of Open Source to gain advantage for proprietary software. In a world in which a technology must be brought to market quickly and just as quickly achieve nearly universal distribution merely in order to survive, there are many products which are stymied by the chicken-and-egg problem: how to get the user or client piece out there, when there are few users of the server component, and, reciprocally, how to get the larger server engine adopted, when it cannot be demonstrated that there are many correspondent client pieces out there waiting to be served. The traditional approach to this is to give the client piece away, hoping that low price will make rapid adoption more likely. The vendor still has the obligation to maintain and distribute the client piece.

I would like to suggest that an Open Source model could be adopted for the client piece. To take a concrete example, the VRML standard for 3D viewing over the Web (currently being renamed and updated into the Web3D standard), depends on users having a VRML browser attached to their Web browser. For a variety of reasons, efforts at developing and distributing VRML-standard browsers have not been successful, and have been largely abandoned. A company wishing to promote a server-side (that is, development) VRML product should release an Open Source VRML browser project. Because the browser would be useful as a reader for all VRML objects, not just those from the particular vendor, the developer community might well take an interest in perfecting and maintaining such a browser, because it would have uses far wider than serving a single vendor. The vendor's benefit would be seeing the VRML market expand, and growing close ties to the VRML developer community that would both provide Open Source support for the browser, and be the likeliest customer pool for the server product.

There is room for all sorts of combinations of profitable cooperation between commercial software vendors and Open Source software—we have hardly begun to try them all.


Layers/Licenses
Operating System === GPL
Toolkits; Extensions
[Free]
Toolkits; Extensions
[Proprietary]
Tools
[Free]
Applications
[Free and Proprietary]
Applications
[Proprietary]
Tools
[Proprietary]