OSBR.CA

The Open Source Business Resource

OCTOBER 2007

OCTOBER 2007

PUBLISHER:
The Open Source Business Resource is a monthly publication of the Talent First Network. Archives are available at the website: http://www.osbr.ca

EDITOR:
Dru Lavigne
dru@osbr.ca

ISSN:
1913-6102

ADVERTISING:
Rowland Few
rowland@osbr.ca

GRAPHICS:
Ryan May

In his book Open Source Licensing: Software Freedom and Intellectual Property Law (http://www.rosenlaw.com/oslbook.htm), Lawrence Rosen defines licensing as "the legal way a copyright and patent owner grants permission to others to use his intellectual property". When you consider that the bread and butter of a company usually revolves around its intellectual property, it's not suprising that open source licenses are often regarded with suspicion. How is it possible for a company's interests to be protected by a license written by another party? And how can a company provide "open" access to its intellectual property without "giving away the store"?

Simon Phipps, Chief Open Source Officer for Sun Microsystems, once stated in an interview: "While open source licensing lets people have access...this doesn't have to mean that chaos ensues." This issue of the OSBR provides insights to help navigate the chaos that is often associated with open source licenses.

We're pleased to include articles from two lawyers specializing in technology law. Lawrence Rosen, quoted above, describes the new QNX hybrid licensing model which is intended to meet the needs of embedded systems developers within the QNX ecosystem. While this model does not meet the requirements of the Open Source Definition, its goals and processes will be familiar to anyone involved in open source. Thomas Prowse draws upon his experience with corporate clients to provide a practical framework for managing open source licenses.

While working with enterprise customers, Stormy Peters from OpenLogic was surprised to discover that the licenses used by their customers differed from the usage statistics commonly encountered in the media. Her article also provides an overview of enterprise best practices.

Kamal Hassin provides an overview of case law applied to open source licensing and Bruce Montague describes the origins of the BSD and GPL licenses, their intents, as well as their advantages and disadvantages.

Finally, two project leaders describe the process they used to determine which license best suited their needs and what they learned along the way. While both projects happened to select the same license, the decision making process may lead other projects with different goals to decide upon a different license.

As always, we look forward to your feedback. Let us know about your licensing experiences by sending an email to the Editor (dru@osbr.ca). We'll publish those of interest to OSBR readers in next month's Letters to the Editor section.

Dru Lavigne,

Editor-in-Chief

Dru Lavigne is a technical writer and IT consultant who has been active with open source communities since the mid-1990s. She writes regularly for O'Reilly and DNSStuff.com and is author of the books BSD Hacks and The Best of FreeBSD Basics.

"It's time to tear the wall down. For too long, an onerous and obsolete barrier has divided the worlds of proprietary and open source software. A barrier that has forced developers to choose between one world or the other — when they could be enjoying benefits of both."

QNX website

QNX Software Systems' new software model integrates open source and proprietary software products in new ways. It is a step forward in the embedded systems market toward openness and freedom of software development, and it gives QNX customers significantly greater flexibility to extend and adapt QNX technology for their own purposes.

The new QNX model is an effort to address fundamental problems in the way proprietary embedded software is traditionally developed and distributed. Today, the rate of change to software and hardware is so rapid, and software so complex, that vendors and customers alike struggle to keep up. Often, software vendors are their own worst bottleneck, as they work to fix or extend their existing products while also attempting to satisfy new, and often divergent, customer needs. Meanwhile, the sophisticated users and customers in the embedded market often know exactly what features and functions they want; many would make the modifications themselves if allowed to do so. And many of them would welcome opportunities to cooperate and share the results of their collective development efforts, just as they would in an open source project.

A pure open source approach doesn't work in all cases, and it doesn't work for QNX which does not believe that relinquishing all control over their intellectual property and giving it away for free would best serve the interests of their customers.

Technology companies implement their fundamental business strategies through licensing their intellectual property. It is a subtle task. If a company gives too much away through overly generous grants of copyrights or patents, then its competitors and customers get a free ride on its products and the company loses its incentive to invest in research and development. If the company makes restrictions on use too tight and complicated, it discourages customers from taking advantage of what its products have to offer. This is where QNX is looking to innovate, with a new blend of transparent development and accessible licenses for the embedded development community.

The goals of open source, built upon licenses that promise freedom to use, copy, modify, and distribute the software that people receive, are becoming part of the nature of the entire software and technology industry. Customers and vendors alike demand open source advantages to be included in their software-based products.

That is what QNX is doing: offering their development community the freedom to proceed without the company being the bottleneck. It is an enablement strategy that combines the benefits of an open source development model with the sustainability of a royalty-based business model for commercial projects. It isn't entirely open source; rather, it's a mixture of open source and proprietary software and rules. I'll explain why.

What's Already Open and What Isn't?

There is already much open source software inside QNX runtime technology and associated development tools. The varied and growing list of the open source components of QNX software is published at http://licensing.qnx.com/license-guide/.

QNX licensees are encouraged to take those open source components and do with them whatever their open source licenses allow.

In addition to incorporating open source software into its products, QNX also serves as a major contributor to open source software. QNX was a founding member of Eclipse and continues to manage the Eclipse C/C++ Developer Tools (CDT) which is based on code that QNX donated to Eclipse.

QNX has released major components of its own software under open source licenses and will continue to release more over time. Many of the QNX board support packages (BSPs) are now available under the Apache License, Version 2.0. This is key to extending the amount of usable hardware available for QNX applications and will enable users to build their own BSPs to satisfy their own needs or those of other QNX users. QNX deliberately chose the Apache 2.0 license for this code in order to give developers the option to offer their derivative works for free or for a fee. While Apache 2.0 doesn't force developers to publish their derivative source code, it does provide a framework for open cooperative development.

Meanwhile, key proprietary components of the copyrighted and patented technology at the heart of QNX runtime software remain available only to QNX licensees, as are certain value-added features of the QNX developer tools. The public cannot freely copy, modify, or distribute QNX software, except for the specific open source components within it. QNX software as a whole, meaning the QNX® Momentics® development tools, the QNX Neutrino® RTOS (Real Time Operating System), and a variety of middleware, is available for use only by QNX licensees and cannot be redistributed to third parties without QNX permission.

However, QNX is offering more visibility into its development process and is granting developers more freedom to modify, enhance, and share licensed copies of QNX software and to create new applications around QNX software for their own purposes. Building on its Eclipse experience, QNX has started to publish the source code for key parts of its runtime products and will conduct ongoing product development for those products in the open. Non-commercial development licenses for the full-blown commercial version of the QNX development suite, which includes the QNX Momentics development tools and the QNX Neutrino RTOS, are available for free. Partner licenses are also available at no charge for anyone looking to add their products to the QNX ecosystem.

QNX is, in effect, creating an open source community within its existing and growing community of RTOS, middleware, and development tool licensees. As a result, anyone interested in QNX technology can now cooperate on development for the benefit of the community as a whole. At the same time, by publishing its QNX Neutrino RTOS source code and by licensing its BSPs under Apache 2.0, QNX is inviting others to take the powerful QNX technology platform down new open or commercial development paths. QNX has even created opportunities where commercial developers can implement and promote the use of QNX technologies for use with target operating systems other than the QNX Neutrino RTOS, and is prepared to license its proprietary technology for those purposes.

To enable these activities, QNX intends to eventually publish all of its runtime component source code and to let developers use that code to create derivative works. As this source code is published, the associated product development activity will also be moved into the public arena.

Traditional open source communities are open to anyone who wants to participate and to follow community rules of behavior and licensing. This QNX community is similar, but the laws of intellectual property, and the limitations that QNX places on the use and distribution of its copyrighted and patented software products, gives this community more of a commercial feel and practice. Anyone can join, and they can become QNX licensees for free as long as they promise not to license their QNX or derivative work software to third parties who aren't also QNX licensees, unless they get a commercial distribution license from QNX.

This community consists only of QNX licensees; that isn't open source, but it is a realistic modification of open source rules to create an open development community for QNX software. Outside of the community of QNX licensees, QNX proprietary software is published but it isn't open. Within the QNX community, developers enjoy the benefits they would find in an open source development environment while at the same time still being able to leverage the advantages available to those who use proprietary products.

The QNX Development Community

Open source software thrives when a community of users and developers cooperate to develop new solutions for the entire community to share. There are many successful open source projects that work on common goals, exchanging ideas and code, mentoring and motivating each other, building product expertise, forming partnerships, and profiting from their collective work. QNX wants its software to grow through that kind of community effort.

Perhaps the most important aspect of the new QNX strategy is the creation of a user and developer community that is internally open and sharing, even though parts of it remain closed to those who don't license QNX software. Within the community, developers can find all of the beneficial aspects of open source development, including transparency of the contribution process, visibility to priorities and projects, merit-based community collaboration, and freely available development tools and resources.

To provide access to developer and customer resources relating to the QNX Neutrino RTOS and the QNX Momentics development tools, QNX is launching a community-oriented web portal called Foundry27 (http://community.qnx.com). Anyone can access information from QNX and from others in the community about QNX products and services, including all published source code.

As the commercial entity that will provide the resources and website upon which the community will work and share, QNX will also help the community with the basic maintenance and coordination needed to sustain a healthy environment. QNX will publish a development roadmap, take steps to evaluate contributions, verify the provenance of contributions provided for adoption into the head branch, and provide infrastructure support through the portal for the benefit of the entire community.

Within the QNX community, developers are encouraged to share their modifications to QNX code with one another for experimental and commercial uses. If their derivative works are based on proprietary QNX code, they can do so provided their code is offered only in substitution for the original form of the work licensed directly from QNX Software Systems and only as long as that original work remains licensed from QNX.

This substitution concept enables others to share modifications to what remains an underlying QNX proprietary code base. QNX also encourages the sharing of modifications to code that QNX has published under Apache 2.0.

Three Classes

QNX has created an innovative way to enable free sharing of derivative works based on either open source or proprietary QNX code. The new hybrid software model divides QNX products into three classes:

1. The first class of software is a small set of patented or copyrighted proprietary QNX software that is based on unpublished source code. Soon this will be limited to certain QNX value-added tools and some QNX middleware products.

2. The second class is a large set of published source code for proprietary components of QNX software that is available for the creation and sharing of derivative works within the QNX community.

3. The third class is a large collection of published source code that is available from QNX under open source license terms, or that has been made available for free from other members of the QNX community to satisfy customer and community needs.

Derivative works from community members and executables built for target systems based on the QNX Neutrino RTOS will often depend on software in the first or second classes, and to that extent companies will need a commercial patent and/or copyright license from QNX to distribute such products.

Deciding what software goes into what class is a balancing act. Claiming too many intellectual property rights for QNX will limit the ultimate success of the community that QNX hopes to empower. The balance is maintained by the company's commitment to publish more of its software over time, and by its promise to cooperate to allow customers and the development community greater creative and licensing freedom with QNX software.

QNX Neutrino RTOS runtime technologies and the QNX Momentics development tools aren't open source in the way that the Open Source Definition (http://opensource.org/docs/osd) requires, and don't claim to be. But the QNX approach to enabling the sharing of derivative works within the community is open source, and is familiar to anyone who has received and used open source software.

This new, more transparent development model serves the needs of QNX customers without giving away QNX's valuable copyrighted and patented technology for free. It incorporates the advantages of the open source development and distribution model, but strictly within the community of QNX Neutrino RTOS licensees who are themselves licensed to benefit from shared advances in the QNX software technology.

New Licenses to Set Community Rules of Sharing

When developers download the QNX Momentics development suite, they can choose from one of three QNX licenses, the first two of which are free of charge:

1. Non-commercial end users: licensees may receive the QNX development suite, under a royalty-free QNX Non-Commercial End User License Agreement (EULA), for certain evaluation and limited development purposes.

This EULA is intended for individuals or companies to experiment with QNX software and to prototype target systems.

2. QNX community partners: licensees may become QNX technology partners to offer their own products and services to QNX customers. QNX now offers its technology partners the QNX Partner Software License Agreement (PSLA) at no charge.

3. QNX commercial customers: the most important participants in the QNX development community are the companies developing commercial applications under the QNX Commercial Software License Agreement (CSLA). This development license isn't free; it includes certain warranties and indemnities by QNX that are appropriate for commercial software. A Standard Support Addendum is included in the CSLA; commercial customers can also benefit from enhanced QNX support with an optional Priority Support Agreement. Licensees will need to execute a separate QNX OEM License Agreement or QNX Runtime License Agreement in order to manufacture and distribute target systems that embed the QNX Neutrino RTOS software.

All of these licenses authorize participation in the QNX development community and allow developers to develop derivative works of QNX software that can be distributed to other QNX licensees.

License Keys and Sharing of Software

Access to QNX software and the QNX web portal is still controlled by license certificates and license keys assigned to companies and individuals who expressly accept the software under the terms of the EULA, PSLA, or CSLA.

None of the QNX licenses allow developers to share their license keys with others. Only those who accept the EULA, PSLA, or CSLA are allowed to share software that the community develops.

While coordination at the QNX development portal is encouraged, QNX doesn't mandate that its community portal be the only development and distribution vehicle for QNX-related products. Licensees may participate in other academic development labs or commercial and non-commercial projects, as long as all the participants are themselves licensed by QNX.

It is this unique combination of open development and strict licensing controls that is exciting about the new QNX business model. I'm looking forward to seeing the embedded systems community adopt and enhance their QNX software to meet their own needs, in an open way, while protecting the proprietary components that are at the heart of the QNX embedded systems business.

This article is based on the QNX Whitepaper, "The New QNX Hybrid Software Model" (http://www.qnx.com/download/download/16868/qnx_whitepaper_hybrid_software_model.pdf).

Lawrence Rosen is a founding partner of Rosenlaw & Einschlag, a technology law firm that specializes in intellectual property protection, licensing, and business transactions for technology companies. Larry served as general counsel and secretary of the Open Source Initiative (OSI) and currently advises commercial open source companies and non-profit open source projects, including the Apache Software Foundation. His book, Open Source Licensing: Software Freedom and Intellectual Property Law, was published by Prentice Hall in 2004.

"The term 'holistic' refers to my conviction that what we are concerned with here is the fundamental interconnectedness of all things…. I see the solution to each problem as being detectable in the pattern and web of the whole. The connections between causes and effects are often more subtle and complex than we… might naturally suppose…."

Dirk Gently's Holistic Detective Agency
by Douglas Adams

This article will set out a practical five stage approach to Open Source Software (OSS) legal issues for organizations that are working, or thinking of working, in this area. While OSS affords a plethora of legal challenges and ongoing developments that merit treatment, I will focus on a general framework for managing OSS legal issues. Since I will provide general legal information and not legal advice, I strongly encourage your organization to work with legal counsel with competency in the OSS area to address its specific circumstances.

Stage 1: Organizational Objectives

The first stage is to achieve clarity with respect to your organizational objectives around OSS. It is essential to start here since the set of clear objectives, which will vary from organization to organization, will be the key driver for each of the following stages.

While these objectives will often have a commercial dimension, the objective may sometimes be philosophical or political in nature. For example, a government organization may be attracted by the "green IT" opportunities of an open source thin client architecture or the competitiveness agenda possibilities of a local OSS ecosystem.

On the commercial front, the objectives can vary widely. They can range from cost-avoidance, to liberation from proprietary solutions, to best-in-breed adoption in a mixed software environment of home grown, commercial, and open source software. In some cases, the organization may intend to create a business around OSS. This could be in the form of a service company focused on OSS support, a hosted services offering, or a dual-license play. In all cases, it is important to always question whether OSS provides the best support for the organizational objectives or whether another solution is more appropriate.

Stage 2: Selection

Where your organization has decided to proceed down the OSS path, the second stage is to decide on the OSS solution(s). In doing so, you will need to consider the pedigree of the code and assess any known risks arising from its use. You will also need to give very careful consideration to the OSS license terms with regard to the manner in which you intend to make use of the code. Whether the license is permissive or reciprocal in nature, whether the code will be used internally or for delivery of a hosted service, whether the code will be modified and distributed, and/or whether the code will have an association with proprietary code, will all impact your organization's potential obligations with respect to the code.

As an organization, you will also need to decide whether your OSS selection can be done on an ad hoc basis or whether it is better to put an appropriate approval body in place. In either case, it is essential to assemble a team with the requisite business, technical and legal skill sets needed for the selection process.

In addition, you should formulate an OSS policy to guide your organization's selection process. It is prudent for this policy to also address voluntary contributions by your organization's employees to OSS projects.

Stage 3: Implementation

Once your organization has selected its OSS solution, it needs to proceed to the implementation stage. At this point, very careful consideration needs to be given to the architecture of your organization's offering. While this stage is tightly tied to, and often iterative with respect to, the selection stage, the analysis at this stage is more holistic having regard to the interrelationship of all of the code components. Although this analysis may be fairly simple in some situations, it is often incredibly complex in a mixed software environment. In addition, architectural options may have profound impact on your organization's OSS obligations so a careful consideration at this stage will pay dividends down the road.

A central element of the implementation stage is a consideration of license interaction. Since almost all OSS and commercial software licenses come with certain conditions, requirements, and/or obligations, it is essential to fully understand the interplay of all of these elements with regard to the compatibility of the licenses. In some cases, it may be necessary to revisit the OSS selection stage, due to irresolvable conflict between the licenses for certain selected software components, before you will be able to finalize your organization's offering. License compatibility will be even more complex in the post GPLv3 world, given the wide range of compatibility customization options now available under that license.

Stage 4: Compliance

Now that you have settled on your organization's offering, with its underlying OSS solution(s), you need to focus on compliance matters. In the first instance, you need to ensure that you are fully compliant with the obligations under the applicable OSS and commercial licenses. For most permissive OSS licenses, your only obligation will be the appropriate reproduction of the applicable OSS license notices.

The situation under reciprocal OSS licenses will be more complex. In cases where your organization will be distributing modified code, you will typically be required to make that code available in source code format. In situations where you will be using OSS code in modified form to provide a hosted service without distribution of the modified code, you will typically not be required, under most reciprocal licenses, to make the source code available. The release of source code would, however, be required in this hosted service scenario under the terms of the Affero license (http://www.affero.org/oagpl.html).

The compliance situation for reciprocal OSS licenses is even more complex with respect to certain associations between software solutions. Under the terms of the GPL license, the licensee is required to release the source code for any "work based on the program" that is governed by the GPL license terms. This determination, driven by an analysis of derivative works principles under copyright law, is by its very nature extremely fact specific. Accordingly, it is essential for your organization to work closely with OSS business, technical, and legal experts to arrive at a well thought out position on this issue.

Stage 5: Audit

For those organizations that have completed the four prior stages, the audit stage is primarily focused on verifying compliance with the steps set out for each of the earlier stages. In particular, the focus of the audit is to ensure that the organization is in full compliance with its OSS obligations including the flow-through of OSS license terms and the release of any required source code.

This article has focused on a "green field" OSS program where the organization is starting from scratch and has no existing code base. In other instances, your organization will want to audit its legacy code base to identify any underlying OSS issues. Black Duck Software (http://www.blackducksoftware.com/) provides one of several existing commercial offerings that can assist an organization in conducting this code analysis.

In addition, your organization may need to audit its supply chain with respect to OSS content in third party commercial offerings and take steps to ensure that appropriate controls and contractual provisions are put in place. In non "green field" cases, your organization will need to initially focus much of its effort on the audit and compliance stage before it will be able to transition to the cadence of the five stage approach.

Conclusion

Any organization that is using or considering the use of OSS needs to give careful consideration to each of the five stages set out in this article. Given that the overview of each stage is illustrative only, and not exhaustive, I encourage your organization to remain open to related legal issues that may either be variants of existing issues or new matters.

While working through the “interconnectedness of all things” will no doubt bring its challenges, the five stage approach to OSS legal issues will provide your organization with a practical framework for the responsible use of OSS by allowing your organization to maximize its use of OSS while minimizing the associated legal risks.

Thomas Prowse, a partner with Gowlings' Kanata Technology Law Office, practices in the area of technology law. His private practice, government policy, and in-house counsel experience ground his deep understanding of the business and technological complexities faced by companies today. As Nortel Senior Counsel from 1994 to 2007, Thomas provided general legal support to global product development organizations and worked extensively on Open Source Software matters.

"Most companies will likely find it acceptable to use open source in some form, but just what form that is can vary greatly from company to company. Which licenses are acceptable is one of the things that companies commonly evaluate ... more often than not they come to similar conclusions."

Jason Haislmaier,
Holme, Roberts & Owen LLP

Studies show that most open source projects are licensed under the General Public License (GPL) and it is estimated that over 75% of open source projects are licensed under either the GPL or the LGPL (Lesser GPL). Yet, it has been my company's experience that the open source software used by our enterprise customers is primarily Apache licensed software. This article examines several factors which may shed some light on this disparity, including the issues raised by enterprise customers and the software product selection process used by OpenLogic (http://www.openlogic.com).

Open Source Licensing Issues

When enterprises consider using Open Source Software (OSS) they are often concerned about legal issues. They already know the software is of good quality and has the features they need because their technicians have tested it and are actively asking permission to use the software in production. Before allowing its use, enterprises want to make sure that they are legally allowed to use the OSS, that it won't jeopardize their own software, and that nobody will sue them for using it.

Common concerns cited by enterprises regarding open source licenses include:

- they are relatively new, and therefore an unknown

- they are mostly written by developers instead of attorneys, so they don't use standard, and well understood, legalese

- the meaning of the term "derivative work" isn't clear when applied to software

- until very recently, few had been tested in court so it was anybody's guess as to how they'd be interpreted by the courts

- any dispute resulting in a court case is expensive, regardless of whether you are right or not

How Enterprises Manage Licenses

When encountered with legal worries about OSS, what do enterprises do? They ask their attorneys to review the license. Initially this can seem overwhelming as the OSI (http://www.opensource.org) has approved over 50 licenses as meeting the open source definition and there are many more licenses that haven't been approved. Fortunately, a few licenses, GPL, LGPL, BSD and Apache, are used by most projects.

Enterprises typically review all of the licenses they use, even for a small one-off application. So, while the majority of software might be released under a couple of licenses, those additional licenses still create a lot of work for enterprise attorneys.

For that reason, enterprises often create OSS policies that explicitly state which licenses are allowed and for which use. An example policy may allow GPL licensed software for use within the company but may not allow its use in products shipped to customers.

Here are some of the best practices I have seen enterprises use when it comes to open source licensing:

- creating an open source policy that clearly defines which licenses may be used; very few companies approve a license for general use as how the software is used can change license compliance

- creating an open source review board that reviews and either approves or rejects every use of OSS, taking the license into account

- requiring an attorney, either as part of or separate from the review board process, to review the license of any OSS being evaluated for use

- keeping a central repository of all of the OSS that is approved for use within their company

- identifying an open source champion or "go to" person for all OSS questions

- when acquiring another company, auditing that company's OSS usage and policy before the acquisition

- tracking the OSS for license changes

That last point is important as projects sometimes change the license when they release a new version of their software. For example, when the Apache Foundation moved from the Apache License 1.1 to the Apache License 2.0, they added an anti-patent clause stating cases where users could not sue the creators of the Apache software. I've seen projects move from a non copyleft to a copyleft license (http://www.fsf.org/licensing/essays/copyleft.html/), or from a license containing no anti-patent clauses to a license containing a strong anti-patent clause. These changes can have major implications for enterprises depending on how they are using the software.

Many enterprises also research indemnification options as insurance against liability in possible future legal suits. Enterprises realize that not only are there potential legal issues around OSS, there often is no "throat to choke"; they need to explicitly ask for indemnification for OSS. Due to the scarcity of test cases in the courts, enterprises often want more indemnification for open source software than for the proprietary software they use.

A good policy comes from the realization that you can't eliminate all risk; policies are about mitigating, not eliminating risk.

Licenses Used by Our Enterprises Customers

OpenLogic provides over 300 (http://www.openlogic.com/downloads/OpenLogic.Certified.Library.pdf) customer requested, certified, supported, indemnified, and updated OSS packages to enterprises. We were curious as to which licenses applied to the software our customers most commonly used. We initially assumed the GPL, as the majority of OSS is licensed under the GPL, but decided to check our database of software. Of the 300 OSS packages in our certified library, only 29% are licensed under the GPL or LGPL and 35% are licensed under the Apache license.

It gets even more interesting if you look at just the top 20 most used software. After sorting our library by number of customers using the OSS package, I took the top 20, grouped them by license and found that:

- 75% were Apache licensed

- 20% were licensed under the GPL or LGPL

- 20% used the CPL, Eclipse, Perl, or BSD licenses

There are several points to keep in mind when interpreting these results:

• the percentages add up to more than 100% as several software packages were dual licensed

• OpenLogic provides software not already included in major Linux distributions; these numbers do not reflect the GPL licensed kernel or any included software packages which tend to be GPL licensed

• most results rely on SourceForge (http://sourceforge.net/) data and does not include much of the software used by enterprises such as Apache, Firefox, and OpenOffice

An overview of the OpenLogic software certification process is needed to determine if it introduces any licensing bias. In order to be added to the Certified Library, an OSS product is assessed against several criteria. The software should:

• have broad adoption based on market research

• provide features required by enterprise customers

• have equivalents to provide companies with other open source alternatives

• be requested by enterprises

In addition, each software undergoes five assessments which validate its viability, license, functionality, support, and technical configuration.

Possible Interpretations

So now the interesting question becomes: are our results coincidence or cause and effect?

Are enterprises, or the OpenLogic selection process, consciously choosing Apache licensed software over GPL licensed software, or is there some other phenomenon at work?

Enterprises may prefer the Apache license over the GPL due to the fear that they will unintentionally have to open source their software. This fear is a common myth; any enterprise required to license software under the GPL could just stop using and distributing the GPL licensed software. In all of my conversations with enterprises, I've only run into one that had "an absolutely no GPL software" policy, although several of them have a “no GPL except Linux” policy. But many attorneys I speak to prefer Apache licensed software over GPL licensed software.

The Apache Foundation produces very high quality software. While anybody can create a new project on SourceForge with no review or vetting, creating a project on Apache.org requires following a rigorous process. Finally, many enterprises are doing Java development and many of the Apache projects, like Struts and Tomcat, are geared towards the Java developer.

Stormy Peters is the Director of Community and Partner Programs at OpenLogic. Before joining OpenLogic, Stormy worked at Hewlett-Packard where she founded and managed the Open Source Program Office. As an early adopter of open source, Stormy was responsible for HP's open source strategy, policy and business practices and was a founding member of HP's Linux Division. Stormy is a frequent keynote speaker on business aspects of OSS and has addressed the United Nations, European Union and various U.S. state governments on OSS. Stormy graduated from Rice University with a B.A. in Computer Science.

"Although I am a typical loner in my daily life, my consciousness of belonging to the invisible community of those who strive for truth, beauty, and justice has preserved me from feeling isolated."

Albert Einstein

The way we develop software is continuously evolving: the everyday processes and practices used to produce software are becoming more efficient, and it is common for a team of developers to change several times over the life of a software project and for the components used to come from a variety of sources. However, the benefits of these changes cannot be fully appreciated unless correct policies and strategies are used to capture value from innovation. This is where the worlds of technology and Intellectual Property (IP) law collide and where license compliance is fundamental in protecting a company's IP and avoiding legal conflicts.

Origins

By understanding how the goals and perceptions of licensing have changed over time, we get a clearer picture of the roots of today's IP conflicts. Open source licensing is not a radically new concept in software development, as can be seen by examining the most commonly used open source license, the General Public License (GPL). In 1989, the Free Software Foundation or FSF (http://www.fsf.org/) released the first GPL which contained a statement of purpose and addressed the major issues of selling, copying, and modifying software. However, it was not written in legal terms and was treated as a social contract rather than today's legal document to be debated in courts. The GPL was adopted as a social framework establishing a general set of rules and expectations for authors, users, and co-developers to observe.

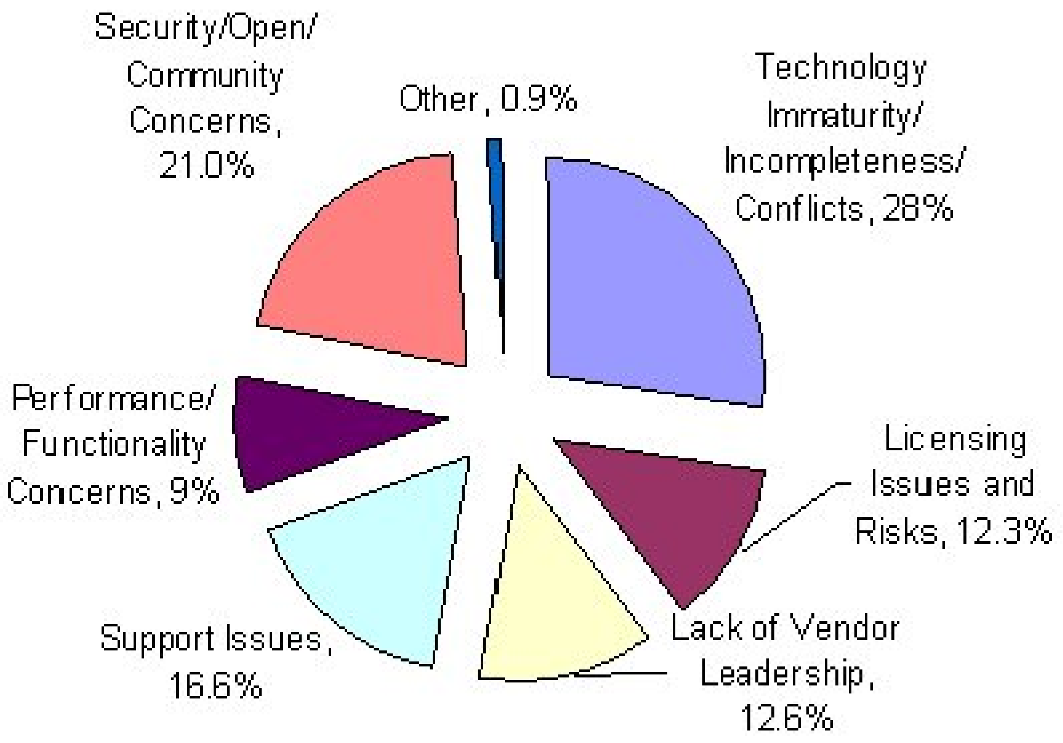
Yet the transformation of this particular license from philosophical theory to its present-day legal document is not always acknowledged in the commercial software industry. As a result, the legal risks and responsibilities associated with the license are sometimes overlooked.

From an IP perspective, using an open source solution is no riskier than using a proprietary software equivalent. From an end-user perspective, the licensing models are similar. Like proprietary software, commercial support contracts for Open Source Software (OSS) usually incorporate some form of indemnification clause to provide protection, usually financial, against potential third-party lawsuits of IP infringement. But there is sometimes the perception that open source is more vulnerable to IP conflicts because it offers indemnification.

It only complicates the situation that the number of open source licenses is increasing and that the licenses are evolving in their legal complexity. According to analyst group Saugatuck Technology, (http://research.saugatech.com/fr/researchalerts/378RA.pdf) there are more than one thousand open source licenses, though most of us only hear about the GPL, BSD, and a handful of others.

IP conflicts and legal compliance issues usually arise when developers and managers fail to address open source licenses in a legal context. As evidenced by the percentages assigned to the factors shown in Figure 1, "Licensing Issues and Risks" is not perceived by organizations to be a primary inhibitor.

Figure 1: Inhibitors to Open Source Adoption by Category
Source: Saugatuck Technology Inc., Worldwide Open Source User Survey, August 2007



Risks and Restrictions

One risk is IP infringement resulting from using unauthorized third-party code or from combining incompatible licenses. For example, a software component licensed under the GPL cannot be distributed with components licensed under the incompatible Mozilla Public License (http://www.gnu.org/philosophy/license-list.html). Yet, with the wide availability of software components, there are multiple opportunities for infringing code to enter a software project. And, many organizations do not have processes in place to catch and address such occurrences.

The typical open source license is designed to protect the contributor of code as opposed to the licensee. This shifting of risk for IP infringement to the licensee is uncommon in proprietary software development; if a software company sells an unclean product, the end-users are not named in an IP infringement lawsuit.

But in the open source world, understanding license obligations and code pedigree is the responsibility of the licensee. And therein lies an important distinction: end users of commercially licensed software are not in anyway liable for the IP integrity of the code, whereas end users of software under an open source license assume the full responsibility for the IP integrity of the code.

Different interpretations of open source licenses have also led to IP conflicts. It is common for open source licenses to use terms that have no precise and agreed to definition. An example commonly used to emphasis this point are the terms "derivative work" and "collective work" which occur in various licenses such as the GPL. There is also the “linking” debate as to how tightly proprietary software can be coupled with GPL licensed software. The GPL is often coined a “viral” license since all derivative works of GPL-licensed software must be released under the GPL.

Landmark Cases

In recent years there have been an increasing number of court cases involving open source licenses. As more of these cases make their way through the legal system and judgements are rendered, a better understanding of how open source licenses are interpreted and enforced by the courts is obtained. These cases further establish precedents that establish the validity and enforceability of open source licenses in subsequent cases.

IP conflicts include violations of open source licenses as well as patent and copyright infringements. In 2004, the non-profit GPL violations organization (http://gpl-violations.org) was launched in Germany by Harald Welte in order to enforce the GPL; it claims to have resolved over 100 cases. Two of its main goals are assisting license holders in legal action against violators and in negotiating settlements with violators out of court. In the United States, the FSF has a similar role, but it only enforces the GPL for software for which it owns the copyrights.

Similar cases occur globally; however, the majority of cases are tried in Germany, partially due to differences between German and US law, such as:

- injunctive enforcement in Germany is easier due to a stricter legal due process; a preliminary injunction can be obtained without giving the defendant the chance to defend itself; the defendant has thirty days from discovery of an infringement to apply for injunctive relief, or the court will send the case to a regular copyright trial which could last for years
- an author of a component within a larger software product can stop the infringer from distributing the entire program, not just the part they own
- a plaintiff in the US seeking a temporary restraining order (TRO) must post bond to compensate the defendant in case the TRO is wrongly issued; this is not the case in Germany

Early Cases

In April 2005, one of the first injunctions was granted against a major privately-held network security software firm when Fortinet was accused of including GPL software in certain products and using encryption techniques to actively hide the usage (http://tinyurl.com/2d9pck). gpl-violations claimed that Fortinet broke the two cardinal rules of the GPL: failure to provide the full source code with the distribution, and failure to provide a copy of the full license text. As a result of the injunction, Fortinet eventually released its source code to the infringing product without charge under the GPL.

Harold Welte explained after the trial, “We are not in any way opposed to the commercial use of Free and Open Source Software and there is no legal risk of using GPL licensed software in commercial products. But vendors have to comply with the license terms, just like they would have to with any other software license agreement” (http://www.out-law.com/page-5620).

Another example is Sitecom, a Dutch firm that uses OSS in its wireless access routers. Sitecom was accused of violating GPL conditions when redistributing their product and the lawsuit was upheld (http://www.jbb.de/judgment_dc_munich_gpl.pdf). This was a significant decision confirming that GPL violations are actionable.

These cases demonstrate that German courts will support aggressive enforcement of the GPL. As a result of these new risks, software should not be developed with disregard to the licensing of its components. Brian Kelly, an IP Partner with Manatt, Phelps & Phillips explains, “Case law interpreting the GPL is both inevitable and useful, because parties are going to end up fighting over ambiguities in the license.” These cases create greater awareness of consequences and emphasize the seriousness of open source licenses in a legal context for all in the software industry.

An implication from the preceding legal cases is that companies may be held liable for license violations in any country, even if the GPL is not enforced in their home country.

Derivative Works

In one of the first open source cases to be debated in the US, courts were asked to evaluate the meaning of a “derivative work” (http://tinyurl.com/2tcqyb). The dispute originated from an agreement which granted NuSphere to non-exclusively market the GPL-licensed MySQL database product. The claim was that NuSphere distributed the product that linked directly to MySQL’s source code without releasing the source code. The key point is that linking to GPL software turns the linked software into a derivative work and all derivative works of GPL software must also be released under the GPL. The judge in this case did not want to create a legal test case and refused to treat it any differently then a trademark dispute. The case was settled out of court, but its arguments raised awareness of the GPL's viral implications: the GPL either bars inclusion of GPL code in proprietary programs or forces derivative works of programs linking to GPL code to be released under the GPL.

Indemnification

The case of The SCO Group v. IBM (http://en.wikipedia.org/wiki/SCO_v._IBM) was a landmark event that increased awareness of the importance of indemnification within the GPL community and to customers using OSS.

Corporations that offer proprietary software, like Microsoft, pay a premium for indemnification protection that is bundled into the cost of the license. But this is not always the case for Linux and other OSS. There are essentially three options for OSS customers:
i) assume the risk and work without indemnification,
ii) use the limited indemnification protection offered by Linux vendors, or
iii) purchase outside indemnification from a firm at a premium.

The Open Innovation Network (OIN) (http://www.openinventionnetwork.com/) is an organization that is garnering support from many companies using open source, such as Google. “Knowing they're protected by the OIN,” Google's Chris DiBona argues, “open source developers are more likely to drive the industry forward.”

The November 2006 agreement between Microsoft and Novell will also work together to improve interoperability between Microsoft software and its open source and standards-based counterparts. This is essentially about indemnification where Microsoft promises not to pursue IP infringement claims against those open source developers and customers who play by its set of rules. One of these rules dictates that customers obtain their Linux from Microsoft’s new partner, Novell. There are signs of improvement as this becomes a driving issue for major standards committees especially in the web services market.

Model Trains

A recent court ruling in the case of Jacobsen v. Katzer has shed light on the key relationships that open source licenses share with patents and trademarks (http://jmri.sourceforge.net/k/docket/158.pdf). The suit involves Jacobsen, a scientist and key member of the open source Java Model Railroad Interface project (http://jmri.sourceforge.net/). The plaintiff alleged copyright violations; Jacobsen argued that the defendants violated copyrights by copying and distributing software without including the attribution required by the open source Artistic License (http://www.opensource.org/licenses/artistic-license-2.0.php). The judge refused to grant an injunction against the copyright infringement. This is the first time a US court has ruled on an injunction request to protect OSS; this decision may or may not create a dangerous precedent for open source licensors looking for injunctions.

The court made two important rulings: i) the Artistic License in question is a contract, and ii) the attribution requirement was a condition of the contract, not a restriction on the scope of the license. By interpreting open source licenses as contracts, the law does not allow for injunctive relief to prevent violators from further infringement. For contract breaches, the remedy is usually monetary damages. However, "assessing damages for use of open source software is difficult because the software is given away free," according to Victoria Hall, attorney for the plaintiff.

Insights

These landmark cases in the interpretation and enforceability of open source licenses highlight the importance of compliance and the consequences of failing to meet licensing terms.

These cases have also created a business opportunity for companies to develop tools that ensure license compliance and solve customer licensing issues.

The software industry now has a clearer picture of the legal implications of open source licensing. As more cases are tried before courts, useful case law will be created to help interpret future conflicts with more certainty. Many companies are implementing policies to verify third-party components used in software projects as failing to do so can result in costly litigation and the remediation and re-engineering of non-compliant software. License compliance is not just a concern for lawyers anymore, but a company-wide undertaking that includes IT staff, software developers, project managers, and executives.

Kamal Hassin received a B.Eng. in electrical engineering from Carleton Univerity in 2004. He is currently a Master's student in Carleton University's Technology Innovation Management program. His research interests include methods to ensure clean intellectual property in software projects, intellectual property law, open source licensing models, and open educational resources.

COMPARING THE BSD AND GPL LICENSES

"Software comes from heaven when you have good hardware."
Ken Olsen, founder of DEC

There are many reasons, not all necessarily altruistic, for the popularity of Open Source Software (OSS). This article provides an overview of software and licensing, and suggests usage examples for two well-known open source licenses: the GPL and BSD license. This article does not discuss recent GPLv3 developments and reflects my own experiences, not necessarily those of my employer.

Origins of Software Licensing

Long before the term open source was used, software was developed by loose associations of programmers and freely exchanged. Starting in the mid 1950's, volunteer user organizations such as SHARE (http://www.share.org/) and DECUS (http://www.encompassus.org/) developed much of the software that companies bundled with the hardware they sold. Anything that reduced software cost and made more programs available made these hardware companies more competitive.

Things changed in the 1960's. In 1965, ADR developed the first licensed software product independent of a hardware company. ADR, competing against a free IBM package originally developed by IBM customers, patented their software in 1968. To stop sharing of their program, they provided it under an equipment lease in which payment was spread over the lifetime of the product. ADR thus retained ownership and could control resale and reuse (http://www.softwarehistory.org/history/Goetz1.html).

In 1969, the US Department of Justice charged IBM with destroying businesses by bundling free software with IBM hardware. As a result of this suit, IBM unbundled its software and software became independent products separate from hardware.

In 1968, Informatics introduced the Mark IV, the first software product to have cumulative sales of 10 million USD. This rapidly established the concept of software as a product, the independent software company, and high rates of return for software. Informatics developed the perpetual license which is now standard throughout the computer industry, wherein ownership is never transferred to the customer.

Origins of the BSD and GPL Licenses

Marshall Kirk McKusick describes the evolution of the Berkeley Software Distribution (BSD) license in "Twenty Years of Berkeley Unix" (http://www.oreilly.com/catalog/opensources/book/kirkmck.html). In summary, the license was intended to allow liberal modification and redistribution terms for Berkeley Unix code. The license required that the source identify the University of California Berkeley (UCB) as copyright holder, that derived products advertise that they were based on UCB code, and that the UCB not be held liable for any damages resulting from the code.

The new BSD license was created in 1999 by the University of California, in response to a request by Richard Stallman to remove the advertising clause. The new BSD license is effectively a statement that the user can do anything with the program or its source, but without warranty and none of the authors has any liability; in other words, the user cannot sue anybody. The license must be kept with the source code, assuring accurate attribution.

In the late 1980s, Richard Stallman became upset when he could not legally add minor improvements to the proprietary system that had replaced the home-grown system at MIT.

Also, many of Stallman's co-workers had left to form companies based on software developed at and licensed by MIT; there appears to have been disagreement over access to the source code for this software. Stallman devised an alternative to the commercial software license and called it the GPL, or General Public License. He also started a non-profit foundation, the Free Software Foundation (FSF) which intended to develop an entire operating system, including all associated software, that would not be subject to proprietary licensing.

The GPL was designed to be the antithesis of the standard proprietary license; it was intended to keep software from becoming proprietary. As the last paragraph of the GPL states: "This General Public License does not permit incorporating your program into proprietary programs" (http://www.gnu.org/licenses/gpl.html).

Open Source Advantages

Open source enables the creation of competitive software that is widely available at the cost of media. Unlike proprietary software, it is not subject to orphaning. Orphaning occurs when a single business failure or change in product strategy causes a pyramid of dependent systems and companies to fail for reasons beyond their control. Decades of experience shows that the momentary size or success of a software company is no guarantee that their software will remain available, as current market conditions and strategies can change rapidly. Since open source development resembles development by an informal consortium, the development team is not dependent on the survival of a single company or product line. Open source licenses and open source projects are the easiest way to form informal consortiums with minimal cost of entry.

Large companies, in which open source code is developed, should be aware that programmers appreciate open source because it leaves the software available to the employee when they change employers. Some companies encourage this behavior as an employment perk, especially when the software involved is not directly strategic. It is, in effect, a front-loaded retirement benefit with potential lost opportunity costs but no direct costs. Encouraging employees to work for peer acclaim outside the company is a cheap portable benefit a company can provide with near zero downside.

GPL: Advantages and Disadvantages

The GPL is a complex license. Here we present some valuable rules of thumb when using the GPL:

- you can charge as much as you want for distributing, supporting, or documenting the software, but you cannot sell the software itself

- if GPL source is required for a program to compile, the program must be under the GPL; linking statically to a GPL library requires a program to be under the GPL

- the GPL requires that any patents associated with GPLed software be licensed for everyone's free use

- aggregating software together, as when multiple programs are put on one disk, does not count as including GPLed programs in non-GPLed programs

- output of a program, such as from the gcc compiler, is not a derivative work

- any code statically linked with the GPLed Linux kernel must be GPLed; this can be circumvented by dynamically linking loadable kernel modules, allowing the use of binary drivers

The GPL is a good choice for code that is intended to remain available to a group of researchers with no future plans for a proprietary fork. The GPL assumes that future scenarios to which a code-base is applicable are understood in advance. Where this becomes an issue, the copyright holder can dual-license the software under both the GPL and another license.

The GPL is attractive to small companies selling CDs in an environment where "buy-low, sell-high" may still give the end-user an inexpensive product. It is also attractive to companies that expect to survive by providing various forms of technical support, including documentation, for the GPLed intellectual property world.

Those who primarily use a system rather than program it or who do not expect to make a living from their work associated with the system find the GPL attractive as it forces code developed by others to be given to them and keeps their employer from retaining copyright and thus potentially orphaning the software. If you want to force your competitors to help you, the GPL is attractive.

For those who must work with statically-linked implementations of multiple software standards, the GPL minimizes the number of programs that can be built because it precludes using proprietary implementations of the standards. A true technical standard should not mandate exclusion of implementations of other standards for non-technical reasons.

The GPL attempts to make programmers contribute to an evolving suite of programs, then to compete in the distribution and support of this suite. This is unrealistic for many standards, which may be applied in varying environments requiring commercial customization or integration with legacy standards under non-GPL licenses.

A less publicized and unintended use of the GPL is that it is favourable to large companies that want to undercut software companies. In other words, the GPL is well suited for use as a marketing weapon, potentially reducing overall economic benefit and contributing to monopolistic behavior. Small companies that are targeted can readily be put out of business.

As intended, the GPL can present a real problem for those wishing to commercialize and profit from software as the GPL was designed to keep research results from transitioning to proprietary products. This step is often assumed to be the last step in the traditional technology transfer pipeline and it is usually difficult under the best of circumstances.

For example, the GPL adds to the difficulty a graduate student will have in directly forming a company to commercialize his research results. An assumption often encountered is that software has become a low-cost commodity; to have significant value it needs to be packaged into a device or a service. A student who has spent years developing a research program might not wish to consider it a commodity.

The GPL is an attempt to keep efforts, regardless of demand, at the research and development stages. This maximizes the benefits to researchers and developers, at an unknown cost to those who would benefit from wider distribution.

Use of a GPL code-base constantly raises commercialization and legal issues. Lawyers working with the GPL have described it as "essentially a full employment guarantee for intellectual property lawyers" (http://p2pnet.net/story/11803).

BSD: Advantages and Disadvantages

The BSD license is intended to encourage product commercialization. BSD-licensed code can be sold or included in proprietary products without restriction on future behavior. It is possible to use BSD-licensed code in GPL-licensed code, but the reverse is not the case. However, do not confuse the BSD license with "public domain"; while an item in the public domain is also free for all to use, it has no owner.

A BSD license is a good choice for long duration research projects that permit anyone to retain the option of commercializing with minimal legal issues. BSD licenses may be preferable for long-term government research intended to ultimately transfer research results throughout the economy in the most widely-deployed fashion possible.

In many cases, the long-term results of a BSD license more accurately reflect the goals proclaimed in the research charter of universities then what occurs when results are copyrighted or patented and subject to proprietary university licensing. Anecdotal evidence suggests that universities are financially better rewarded in the long run by releasing research results and then appealing to donations from commercially successful alumni who benefited from the released IP.

The question "why should we help our competitors or let them steal our work?" comes up often in relation to a BSD license. However, if one company came to dominate a product niche that others considered strategic, a mini-consortium aimed at reestablishing parity through a BSD-licensed variant would increase market competition and fairness. Each company believes it will profit from some advantage it can provide, while also contributing to economic flexibility and efficiency.

Companies recognize the value of de facto standards as a marketing technique. The BSD license serves this role well, for companies with a unique advantage in evolving the system. Sometimes the GPL may be appropriate for a standard, especially when attempting to undermine or co-opt others. The GPL, however, penalizes the evolution of that standard, as it promotes a suite. Regardless of the license used, the software will usually devolve to whoever makes the majority of the engineering changes and most understands the state of the system.

To minimize software engineering problems, such as mixing code under different licenses, BSD licenses should be encouraged. Being leery of the GPL should particularly be the case with non-profits that interact with the developing world. In locales where application of law becomes a costly exercise, the simplicity of the BSD license is of considerable advantage.

Conclusion

There are distinct advantages and disadvantages inherent in any license; this article outlined some usage scenarios for the GPL and BSD licenses. The GPL, while designed to prevent the proprietary commercialization of open source code, can still provide strategic advantage to a company. The BSD license, by placing minimal restrictions on future behavior, allows code to remain open source or become integrated into commercial solutions, as a project's or company's needs change.

Bruce Montague has over 30 years of experience as an OS engineer. He has been a civilian USAF computer scientist, has been on the staff of the US Naval Postgraduate School, was a senior engineer at Digital Research, Inc., has been a developer of filesystems, and has developed a number of custom operating systems, including the first embedded Java OS. He has a PhD in Computer Science from UCSC and currently works for Symantec Research Labs.

"In short, open source is here to stay. It's already had major impact, but there's more to come. Keep your eyes open, and prepare for more positive surprises!"
Tim O'Reilly, CEO, O'Reilly Media

On June 8th, 2005, we officially launched the ePresence (http://epresence.tv/) Interactive Media Open Source Consortium, at the Knowledge Media Design Institute (KMDI), University of Toronto (UofT). We had been researching and developing ePresence, our webcasting, webconferencing, and archiving software project for about five years. Throughout the early phase of the project we used the system to produce live webcasts of KMDI's annual lecture series. Eventually word spread about our webcasting system and other universities, such as Memorial University in Newfoundland, became interested. It was obvious that the time to share our project with the world had come, but what wasn't obvious to us at the time was how we were going to do that.

Why Dual License?

We have always maintained that universities should support open source licensing and knew this was the option we were going to pursue. However, weren't sure which of the many open source licenses available would best suit the project. Because we planned to launch the open source consortium from within the university, we also needed to develop a revenue model. We were asking ourselves the same question everyone must ask when they arrive at this juncture, "how do we make money when we're giving away our software?"

Eventually we decided to split the system into two software products, ePresence Media and ePresence Live!, and distribute each product under its own license.

ePresence Media represents the core of the ePresence system and allows users to record web seminars, presentations or lectures and publish them to the web. ePresence Live!, when used with ePresence Media, allows users to stream content live over the internet.

The rationale behind this dual license strategy was two-fold. It would allow us to release ePresence Media under a BSD license to provide for free availability and use. At the same time, we would release ePresence Live! under a University of Toronto community source license and offer it as one of the benefits of joining the ePresence consortium. By wrapping the live streaming components in a membership package with support and various other benefits, we believed we had created a product that we could market and sell.

It is important to note that under the UofT community license, the source code for ePresence Live! is available to purchasers of membership packages. Our goal was to create an incentive for users to purchase a membership, not to keep the source code closed.

Some might say we were being cautious, others might say we were trying to have our cake and eat it too. Either way, we had to prove to the university and ourselves that we had a model that was capable of generating revenue.

Lessons Learned

At first, the dual licensing strategy seemed to work. But as ePresence grew in popularity, problems with the strategy began to emerge. First, it wasn't the easiest arrangement to explain to potential customers. Part of the problem was the membership agreement was too long.

Another problem was that it included a clause intended to encourage entrepreneurship and redistribution of the software. However, this clause only confused the issue of distribution. Most of our early inquiries were from academic institutions who simply wanted to set up webcast production stations in a couple of locations on their respective campuses, not redistribute the software in a way intended to generate revenue.

However, the real problem of maintaining this strategy emerged from the development side. After a year or so under the dual strategy we soon realized the constraints of developing, testing, and packaging two separate but related software packages. Each time we released a version of the software, we had to go through the steps twice. We were also beginning to utilize other open source applications for ePresence development and managing licensing compatibility was becoming time consuming. But the most interesting and unanticipated problem that emerged from our decision to employ a dual license strategy was one that involved usability.

It wasn't until we began to accumulate more ePresence users that we began to truly understand the learning complexities involved in using the system. We quickly realized that we had to make the system easier to use and with each subsequent release complexity problems were addressed and resolved.

But it wasn't until we understood the learning complexities of ePresence that we began to realize that our decision to implement the dual licensing business strategy had inadvertently introduced a usability problem into the system. The dual license strategy created an obstacle for users simply because it required users to run several interfaces at the same time.

If an ePresence user wanted to stream an event live and capture that content for archive publishing later, that user would have to open an application for each of the streaming formats, plus one for the archive capturing. We needed to take these interfaces and simplify them into one, easy to use interface.

Clearly, the only way for us to do this was to put the system back together and release it as a complete set of webcasting and archiving tools. It also helped that by the time we were ready to rethink our business strategy, processing power had emerged to the point where we could run all of the ePresence applications on one machine.

Relicensing

It was almost as if we had arrived back at square one: we had to decide under which of the two licenses, the BSD or the UofT community license, we were going to release the software. Actually, it wasn't much of a decision at all; we knew if we were going to be viewed as a legitimate open source project then we were going to have to continue with the BSD license. By this time we had added hardware and hosting services to our list of services and products and were feeling more confident in the system and our ability to generate revenue.

On August 2nd, 2007 we released ePresence version 4.0 under the BSD license. Accompanying this release was the revised revenue model that offers five support packages, hardware, hosting and our new community media portal, ePresenceTV. Not only does ePresence offer a set of tools and services that compare to similar propriety products, ePresence is the world's first open source webcasting, webconferencing and archiving software system.

Although it has been only a couple of months since we officially released the software and launched the new support subscription offerings, the feedback thus far has been very positive. We have noticed that bloggers are taking note of ePresence; we have also increased traffic to our website and seen a great improvement in our SourceForge ranking.

In June 2007, at our Annual General Meeting for ePresence consortium members, we distributed an informal survey asking members for their feedback and comments. All of the respondents agreed that releasing the entire system under the BSD license was a good idea and that having the system completely open would be a benefit to adopters. Members also indicated their willingness to remain members of the consortium, and to this date all members who had joined the consortium under the original agreement have renewed their memberships.

Conclusion

By modifying our business strategy and releasing ePresence under a single open source license, we have simplified our sales process by removing the focus from having to explain the complex dual license strategy to putting it where it belongs, on the software's robust functionality, and the products and services available.

Tim O'Reilly warns that there are more open source projects to come and to: "Keep your eyes open, and prepare for more positive surprises!" We think ePresence is one such project, and while it's too soon to declare our venture a success, we are very pleased with the early results, and would have to consider ourselves among the positively surprised.

We would like to acknowledge the Network for Effective Collaboration Technologies through Advanced Research (NECTAR) Network Grant from the Natural Sciences and Engineering Council (NSERC) of Canada, for funding, in part, ePresence research.

Kelly Rankin is the Manager for the ePresence Open Source Consortium. She has given numerous ePresence demonstrations and has produced a number of webcast learning events, including, "The Business of Software" and the Project Open Source | Open Access keynote address. In addition to her activities as Consortium Manager, she is completing her Bachelor's Degree in Philosophy at the University of Toronto.

Ronald Baecker is Professor of Computer Science, Bell University Laboratories Chair in Human-Computer Interaction, and founder and Chief Scientist of the Knowledge Media Design Institute at the University of Toronto. He is Principal Investigator of the Canada-wide NSERC Network for Effective Collaboration Technologies through Advanced Research (NECTAR) and Project Director, ePresence Interactive Media. Baecker is an active researcher, lecturer, and consultant on human-computer interaction and user interface design, and software entrepreneurship.

"It is a common delusion that you make things better by talking about them."
Dame Rose Macaulay, English novelist

Open source provides an avenue for distributing academic research well beyond the covers of journals or the lunchtime chatter of sharp-minded thinkers to a much broader audience. Interestingly, the choice of open source license is often a choice of community. By understanding the goals and underlying philosophy of a research project, one is better equipped to find a suitable license and attract a community with similar interests.

This article provides an examination of a particular academic research project's licensing goals and presents some of the lessons learned during the license selection process.

Why Open Source?

The Nunaliit (http://nunaliit.org) project is a software framework for producing web atlases. From the start of the project, there were many reasons to release the software under an open source license:

- the project leads were already proponents of Open Source Software (OSS) and the idea of contributing back to the community was appealing
- the intention was to incorporate other people's open source code where it made sense
- attracting interest to help develop code was a goal
- open formats are often best supported by open source efforts
- the use of open standards and open source meant that atlases created with the framework would have a better chance at retaining their value to the world over time
- traction with communities, research partners, and funding organizations would be better if we weren't trying to promote proprietary software

In addition, the research was funded by taxpayers and the lab members felt that outputs should be fully accessible to the public; while academic papers are expected by funding partners, there is also value in the process and tools built to prove the points. With open source, the mark of our success could be measured by our ideas being widely accepted, adopted, and responsible for change for the better.

Another factor was the research itself which was aimed at helping communities to tell their stories in new and innovative ways. These were often communities whose voice was not being heard, in large part due to the financial resources available to them. Building a free and open framework meant they weren't dependent on the project in order to use or improve upon the software in the long run.

Which License?

Since building a developer community around the framework was a primary goal, the license and contribution agreements would impact on the success of recruiting people to the project.

A secondary goal was choosing a license familiar to other people, meaning we didn't want to create a custom license. For this reason, the Nunaliit project compared the three best known licenses, the General Public License (GPL), Apache Public License (APL), and New BSD License (BSD), to the type of community each license was likely to attract.

The most troubling issue with the GPL was that it requires all distributed derivative works to be released with the same open terms.

We were not opposed to closed ventures; in fact, if companies were able to make use of our software in their products, they validated our ideas. Likewise, if we ever wanted to commercialize the work in some fashion, we would not want to offend community contributors by dual-licensing the code.

In order to preserve this possibility with the GPL, the necessary over-reaching contribution agreements might have scared people away. Avoiding the possibility of going back to closed software products is, after all, a major philosophy of the GPL.

The Apache and Mozilla style licenses didn't have that same "keep drinking the kool-aid" clause and were seriously considered. But ultimately, they still placed a more significant burden on people who wanted to make use of our code and would require a more substantial contribution agreement.

The BSD license left things wide open. Asking people to contribute wouldn't require them to go for outside legal help to understand what they were doing by putting their code under that license. It was clearly one of the most open licenses, but left the question "would people bother helping the project or would the code just get picked up by some company and improved internally without contributing back?".

A good chat with a friend who has been involved with the Mozilla project since its inception helped to answer that question. The insight that came out of that conversation was that forcing openness in the license has very little to do with whether or not you will get contributions back. Community has much more to do with a project's support infrastructure and its responsiveness to contributions.

If it's an easy and timely process for someone to ask a question, file a bug, submit a patch, and see the result incorporated, they will do so as it's far easier to contribute to an existing project than to maintain a separate fork of the code.

This friend, who has spent a fair bit of time discussing the three separate licenses that Mozilla is released under, suggested that if he was in a position to do it all over again, he would likely advocate for the BSD license to save a whole lot of hassle.

This made a lot of sense for a very small project with limited resources. With the consent of our existing code contributors, the New BSD License was chosen and all existing code was placed under that license.

Even though the BSD license is wide open and the published code is entirely free and open for any use, our project has decided to not incorporate code from projects that have chosen the GPL. This is due to the project's philosophy that BSD licensed software is free (adjective) while GPL software is on a mission to free (verb) software.

Insights

Prior to selecting the license, the project understood that the chosen license would have an impact on potential contributors. Since releasing the code under a BSD license, the following behaviours have been noted:

- contributors tend to select projects that utilize their preferred license

- contributors are also attracted to projects containing technology that matches their interest and skill set

- license selection should consider both the characteristics of the project's technology and the licenses already being used by technologically similar projects

That last point was unanticipated. As a server-side publishing-infrastructure-like technology, Nunaliit may have drawn a bit more interest and understanding by selecting an Apache license.

Conclusion

When evaluating which license to adopt, consider the projects that most closely resemble yours or whose choice of implementation technology is similar. Developers of these projects may be more familiar or even philosophically attached to one license over another and more apt to contribute if your license matches.

Projects should also give serious thought to their motivations and hopes for releasing code to the world before settling on a license.

Recommended Reading

St. Laurent, Andrew, M., Understanding Open Source & Free Software Licensing (http://www.oreilly.com/catalog/osfreesoft/book/).

Chen, Shun-Ling, Free and Open Source Software Licensing Primer (http://www.iosn.net/licensing/foss-licensing-primer/).

Amos Hayes is a technical specialist turned researcher and manager at the Geomatics and Cartographic Research Centre (http://gcrc.carleton.ca) at Carleton University. A good part of his work is to help turn the ideas of researchers from a whole host of different academic disciplines into a set of technical capabilities for an open source community atlas framework.

Q. I've read that commercialization has both a supply and a demand side. What effect do these two sides have on open source commercialization, specifically in Canada?

A. The seminal document covering the state of commercialization in Canada today is “People and Excellence: the Heart of Successful Commercialization” (http://strategis.ic.gc.ca/epic/site/epc-gdc.nsf/en/h_tq00013e.html) written by the expert panel on commercialization. The panel has taken a balanced approach to assessing the current situation and formulating a number of recommendations for improvements in commercialization in Canada. The committee has also gone where few Canadians have gone before by looking at commercialization in a holistic sense where commercialization is the sum of its parts; the two parts of the commercialization puzzle are the supply side and the demand side.

I like to use the mousetrap analogy when talking about commercialization's supply and demand sides. The supply side is all of the ingredients necessary to build the mousetrap whereas the demand side is the ingredients necessary to achieve marketplace success with that mousetrap. Supply side commercialization includes public and privately funded research which generates product ideas and the product itself. Demand side commercialization is all about business models, strategy and market place implementation. Both supply and demand side are essential for successful commercialization.

The Conference Board of Canada in their 2007 “How Canada Performs: A Report Card on Canada” gives Canada a D in innovation and cites our lack of ability to commercialize as a key contributing factor (http://www.conferenceboard.ca/documents.asp?rnext=2047).

Canada is generally acknowledged as doing well at the supply side of commercialization; however, we are notoriously less proficient at marketplace success with the demand side.

I believe the root cause of Canada’s lack of commercialization excellence is related to the Canadian commercialization paradigm. This isn’t to say that there aren’t Canadian success stories; however, on average, Canadian companies underperform most OECD nations in commercialization. A telling sign is the ever widening productivity gap between Canada and the US. For many generations, Canadians have placed maximum emphasis for commercialization success on building the mousetrap while minimizing or ignoring demand side commercialization.

Two indicators of the current supply side paradigm are commercialization incentives and linear commercialization. Commercialization incentives are essential for rewarding the behaviours the government wants to encourage. There are a number of government programs targeted at supply side commercialization; two examples are SR&ED and IRAP (http://www.cra-arc.gc.ca/taxcredit/sred/menu-e.html and (http://irap-pari.nrc-cnrc.gc.ca/main_e.html).

While there are a number of government programs geared toward building mousetraps, there are far fewer incentives targeting demand side commercialization excellence. But, virtually all other sources of funding such as angels, Venture Capitalists, and junior public markets, prefer that the funds are used for marketing an existing product. Research and Development (R&D) is considered too risky!

Linear commercialization views supply and demand side as separate components. The typical linear commercialization sequence involves bright young engineers developing a leading edge mousetrap. The focus right from the start is on R&D excellence and building the best darn mousetrap in the world. Upon completion there is an innovative mousetrap but the world has not beaten a path to the company's door; the next logical step is that the young company switches focus from supply to demand side commercialization. The net result is supply and demand is completed in a linear fashion rather than in parallel.

How does the problem of a supply side centric commercialization paradigm affect an open source business? Well, your choice of commercialization paradigm will have a direct and more significant impact on your open source business than likely any other single factor. In fact, choosing the commercialization paradigm for your open source project will be one of the most important decisions that you make as a business. If you apply the commercialization supply and demand side model to an open source model, your supply side is primarily the code you are developing, while the demand side is the business model you choose. In order to succeed in commercializing your open source assets you will need a paradigm that balances both sides of commercialization.

If you look at some of the early entrants into the open source business market, you'll see companies who were all about passion, code, and supply side commercialization. Marc Fleury, creator of JBoss, stated for BusinessWeek: (http://tinyurl.com/7wglx) "The origin of open-source was definitely non-profit, right? It was very high on passion and church, but not at all with a business model behind it".

Marc Fleury quickly realized the need to develop a balanced commercialization paradigm for his organization that included a viable business model. One of the first incarnations of a business model for open source was to give the software away and charge for service. Rather simplistic, but it did work for JBoss. There are many more business models available to open source today, some of which are described in "Seven Open Source Business Strategies for Competitive Advantage" (http://www.itmanagersjournal.com/feature/314).

Perhaps one of the best studies in a balanced open source commercialization paradigm in Canada, and the world, is the Ottawa-based Eclipse Foundation. This organization started with a balanced commercialization paradigm which, summed up in a word, is collaboration. While many open source organizations foster collaboration in the development of code, The Eclipse Foundation has taken this a step further.

Mike Milinkovich, Executive Director of the Eclipse Foundation explains: "Eclipse has a corporate membership model that has resources (16 full-time staff) to help proactively foster collaboration. Also, the way we're set up and the way our organization is defined, we're explicitly set up and tasked with fostering collaboration and commercial adoption of our products (and commercialization of the application written on top of what we do)" (http://blogs.cnet.com/8301-13505_1-9760440-16.html).

The Eclipse Foundation commercialization paradigm permeates everything they do and the collaborative environment applies equally to supply and demand side. The selection of incorporation as a non-profit business is a key enabler to their success.

Mike continues: “You read the literature on ecosystems and there's always this expectation that there's a for-profit organization at its core that essentially makes things work. In the Eclipse context, at its heart is a not-for-profit organization with an open-governance and open-licensing model. Having a not-for-profit organization at the heart of Eclipse makes a big difference.”

The Eclipse Foundation success can be attributed, in my opinion, to their selection right from the start of a balanced commercialization paradigm. To be successful in commercialization of an open source business you must select a balanced commercialization strategy paying attention to both supply and demand side.

Ian Graham is a certified management consultant working with early stage businesses in the Ottawa area. He has a passion for entrepreneurship and volunteers with Junior Achievement at the local high school and is a key contributor to the Ottawa DemoCamp series of events. Ian is a member of the Ottawa eBusiness Cluster (http://www.ebusinesscluster.com/) executive and also chair of the Certified Management Consultants technology committee. He has a technical diploma from Algonquin College and his MBA from the University of Ottawa. Ian trains with Bizlaunch (http://www.bizlaunch.ca/) and will be teaching a course in product introduction at Professional Programs at the Sprott School of Business in the fall and winter of 2007/2008.

Q & A
LETTERS TO THE EDITOR

Molly from Malaysia writes: Your Open Source reference to assets that nobody owns is not quite correct. (para 2 in http://www.osbr.ca/opensource.php) There is ownership and the owner allows its use on conditions consistent with the various open source licenses available. I like your first paragraph but people in health care may be concerned about "data" as these refer to patient data which must be separated and kept confidential. I suggest the word "content" may be preferable.

Adrian from Ottawa writes: The OSBR looks like an interesting effort. I took a quick glance and one thing I immediately picked up on was this text on the home page: "Open source refers to assets that nobody owns and anyone can use, modify and distribute as well as the processes used to produce them." There are a number of errors in that statement. People do actually own open source assets. If nobody owned them then they would be in the public domain and that's definitely not the case with open source code. Usually there is shared ownership. And while usually most licenses permit use, modification, distribution, etc. there are often conditions that apply. It might be better to say "(subject to license conditions").

Editor: The text on the website will change shortly.

Recommended Reading:

Canada’s New Government, Mobilizing Science and Technology to Canada’s Advantage

http://www.ic.gc.ca/cmb/welcomeic.nsf/vRTF/PublicationST/$file/S&Tstrategy.pdf

Battle for Open Web Standards Gains Open Source Tools

September 5, Vancouver, BC

ActiveState Software Inc., a leading provider of professional development tools, announced the creation of the Open Komodo Project, a new initiative to create an open source platform that promotes open standards. The Open Komodo Project will fill a need for developer tools in the open web technology stack, furthering web innovation and freedom of choice for developers and end-users. Open Komodo aims to create a framework for client-side web development integrated with Firefox®, Mozilla's free, open source web browser, and based on the award-winning Komodo® IDE, a multi-platform, multi-language IDE for dynamic languages and Ajax technologies. As a first step in the Open Komodo project, ActiveState is open sourcing the browser-side capabilities of Komodo® Edit, a free multi-language editor for dynamic languages based on Komodo® IDE.

http://www.activestate.com/company/newsroom/press/2007_09_05_0

QNX Publishes Neutrino Source Code and Opens Development Process

September 12, Ottawa, ON

In a move that revolutionizes software development practices by combining the best of the open source and commercial software domains, QNX Software Systems today announced that it is opening access to the source code of its QNX® Neutrino® real-time operating system (RTOS) under a new hybrid software licensing arrangement. These changes are part of a new hybrid software model created by QNX that supports the customer's goal of profiting from software while fueling the passion for developing it. Access to QNX source code is free, but commercial deployments of QNX Neutrino runtime components still require royalties, and commercial developers will continue to pay for QNX Momentics® development seats. However, noncommercial developers, academic faculty members, and qualified partners will be given access to QNX development tools and runtime products at no charge.

http://www.qnx.com/news/pr_2471_2.html

OSGeo Journal Publishes Volume 2

September 20, Prince George, BC

The next volume of the OSGeo Journal is now available for download (http://osgeo.org/journal/volume2). This content-packed volume includes several case studies, news items, project introductions, an introduction to topology, perspectives from OSGeo sponsors and much more. You are encouraged to link to this page as well as share this announcement with other professionals who are interested in geospatial topics. We aim to have generally thought-provoking articles as well as open source focused technology discussions.

October 25

Myths About Open Source Licensing

Ottawa, ON

Open Source Software (OSS) licensing is a topic that has evolved considerably since the first OSS licences were drafted – the BSD (early 80's) and the GPL (late 80's). With this evolution came some level of complexity, as well as a number of myths about OSS licensing. A taxonomy of OSS licences to help orient potential creators and users of OSS will be discussed.

http://iit-iti.nrc-cnrc.gc.ca/colloq/0708/07-10-25_e.html

October 25-26

FSOSS07

Toronto, ON

Free Software and Open Source Symposium (FSOSS) is a high-profile event that attracts leaders from industry and the open source community in order to discuss open source issues, learn new technologies, and promote the use of free and open source software. The Symposium is a two-day event aimed at bringing together educators, developers and other interested parties to discuss common free software and open source issues, learn new technologies and to promote the use of free and open source software. At Seneca College, we think free and open source software are real alternatives.

http://fsoss.senecac.on.ca/2007/

November 14

webcom 2007

Montreal, QC

At this conference you will learn more about the impact of Social Web on your marketing strategies, the impact of emerging technologies on Enterprise 2.0, and understand more how these new tools transform communication modes.

http://www.webcom-montreal.com/index.php

November 14 - 15

GIS Day at Carleton

Ottawa, ON

Come and join Carleton University's Department of Geography and Environmental Studies and the Library's Maps, Data and Government Information Centre at GIS Day 2007 (http://www.gisday.com/). Explore Where in the World Carmen Sandiego is Now and discover that GIS and Geomatics are more than a Jeopardy category. On Wednesday, November 15, Carleton University is offering several engaging and interactive activities to showcase geomatics. This one-day showcase will provide interactive demonstrations of the GIS technology, exhibits, industry representation, and displays.

http://www.library.carleton.ca/gis/gisday.html

November 22

Open Source Software: Demystify the GPL

Vancouver, BC

If you use Firefox then you are using Open Source Software. Learn how your business can benefit from Open Source Software in this informative 90 minute seminar. While other software licenses contain limitations and restrictions on their use, GPL licensed Open Source Software is flexible and cost effective.

http://www.e-bc.ca/pages/resources/seminars.php

The goal of the Open Source Business Resource is to provide quality and insightful content regarding the issues relevant to the development and commercialization of open source assets. We believe the best way to achieve this goal is through the contributions and feedback from experts within the business and open source communities.

OSBR readers are looking for practical ideas they can apply within their own organizations. They also appreciate a thorough exploration of the issues and emerging trends surrounding the business of open source. If you are considering contributing an article, start by asking yourself:

1. Does my research or experience provide any new insights or perspectives?

2. Do I often find myself having to explain this topic when I meet people as they are unaware of its relevance?

3. Do I believe that I could have saved myself time, money, and frustration if someone had explained to me the issues surrounding this topic?

4. Am I constantly correcting misconceptions regarding this topic?

5. Am I considered to be an expert in this field? For example, do I present my research or experience at conferences?

If your answer is "yes" to any of these questions, your topic is probably of interest to OSBR readers.

When writing your article, keep the following points in mind:

1. Thoroughly examine the topic; don't leave the reader wishing for more.

2. Know your central theme and stick to it.

3. Demonstrate your depth of understanding for the topic, and that you have considered its benefits, possible outcomes, and applicability.

4. Write in third-person formal style.

These guidelines should assist in the process of translating your expertise into a focused article which adds to the knowledgable resources available through the OSBR.

Upcoming Editorial Themes

November 2007	Support
December 2007	Clean IP
January 2008	Interoperability
February 2008	Data
March 2008	Procurement

Formatting Guidelines:

All contributions are to be submitted in .txt or .rtf format and match the following length guidelines. Formatting should be limited to bolded and italicized text. Formatting is optional and may be edited to match the rest of the publication. Include your email address and daytime phone number should the editor need to contact you regarding your submission. Indicate if your submission has been previously published elsewhere.

Articles: Do not submit articles shorter than 1500 words or longer than 3000 words. If this is your first article, include a 50-75 word biography introducing yourself. Articles should begin with a thought-provoking quotation that matches the spirit of the article. Research the source of your quotation in order to provide proper attribution.

Interviews: Interviews tend to be between 1-2 pages long or 500-1000 words. Include a 50-75 word biography for both the interviewer and each of the interviewee(s).

Newsbytes: Newsbytes should be short and pithy--providing enough information to gain the reader's interest as well as a reference to additional information such as a press release or website. 100-300 words is usually sufficient.

Events: Events should include the date, location, a short description, and the URL for further information. Due to the monthly publication schedule, events should be sent at least 6-8 weeks in advance.

Questions and Feedback: These can range anywhere between a one sentence question up to a 500 word letter to the editor style of feedback. Include a sentence or two introducing yourself.

Copyright:

You retain copyright to your work and grant the Talent First Network permission to publish your submission under a Creative Commons license. The Talent First Network owns the copyright to the collection of works comprising each edition of the OSBR. All content on the OSBR and Talent First Network websites is under the Creative Commons attribution (http://creativecommons.org/licenses/by/3.0/) license which allows for commercial and non-commercial redistribution as well as modifications of the work as long as the copyright holder is attributed.