Natural Resources Canada / Ressources naturelles Canada

CANADIAN GEOSPATIAL DATA INFRASTRUCTURE
INFORMATION PRODUCT 33e

Free and Open Source Software Licensing Primer

GeoConnections

Hickling Arthurs Low Corporation

2012

Canada

ACKNOWLEDGEMENTS

GeoConnections would like to acknowledge the contributions made by Hickling Arthurs and Low Corporation – Ed Kennedy, the Canadian Internet Policy and Public Interest Clinic at the University of Ottawa – David Fewer and Kent Mewhort, for providing research, writing and editing of the Primer. We would also like to thank the FOSS community of practice for their direction, comments and critical contributions. The staff at GeoConnections who provided management, input and direction for the project were Cindy Mitchell and Simon Riopel, in addition Catherine Galiano-Lavallée, Mélodie Bujold, and Jenna Findlay provided assistance in assembling this document.

Table of Contents

1. Preamble

This guide is one in a series of Operational Policy documents being developed by GeoConnections. This guide is intended to inform *CGDI* stakeholders about the nature and scope of Free and Open Source Software (FOSS) licensing and the realities, challenges and good practices of related operational policies.

Burgeoning growth of online geospatial applications, coupled with the growing acceptance, use and development of free and open source software, has revealed a significant gap in operational policy coverage for the Canadian Geospatial Data Infrastructure (CGDI).

Currently there is no commonly accepted guidance for CGDI stakeholders wishing to use, develop and share free and open source software. More specifically, there is little or no guidance available to inform operational policy decisions on the licensing of open source software in the CGDI community. Given the importance of interoperability to our national SDI, and the growing national and international movements towards common and standardized open data licences, it is imperative to consider the interoperability of the tools and technologies that support the CGDI; in this case, free and open source software solutions, and the basis of operational policy required to support them.

The **GeoConnections** *program is a national initiative led by Natural Resources Canada. GeoConnections supports the integration and use of the* **Canadian Geospatial Data Infrastructure (CGDI).**

The **CGDI** *is an on-line resource that improves the sharing, access and use of Canadian geospatial information –information tied to geographic locations in Canada. It helps decision makers from all levels of government, the private sector, non–government organizations and academia make better decisions on social, economic and environmental priorities.*

While GeoConnections remains impartial on the subject of free and open source vs. restricted source software licences, it is important to provide essential information and guidance to CGDI stakeholders to inform operational policies that allow organizations to proceed in a manner best suited to their business needs, and in a way that also promotes the highest level of interoperability within Canada’s SDI community.

This primer will provide CGDI stakeholders with information on how to adopt, incorporate and use free and open source software according to varying licensing terms and conditions, and how to distribute (i.e., licence) a new or modified software resource as free and open source software. It is intended to inform CGDI stakeholders on the importance of interoperability in licensing, and provide them with the information and decision-making tools required to make optimal decisions on policy for free and open source use, development and release.

This guide is designed to provide information on free and open source software licensing, including related legal topics, current at the time of publication, for general informational purposes only. Material found in this guide may not apply to all jurisdictions. GeoConnections is not responsible for the use of any materials or contents of this guide. The contents of this guide do not constitute legal advice and should not be relied upon as such.

2. An Introduction to Free and Open Source Software (FOSS) Licences

2.1 Definition of FOSS

The term "Free and Open Source Software" (FOSS) – otherwise known simply as "Open Source Software" or "Free Software" – refers to computer software applications where the authors or copyright holders release the source code under a licence that authorizes anyone to use the software, modify it, and redistribute it.

The Free Software Foundation (FSF) and the Open Source Initiative (OSI) publish the two most widely accepted definitions of free and open source software. The FSF adopts a principles-based approach that promotes the idea that software should be "free as in freedom", and the FSF prefers the term "free software" rather than "open source software". The FSF's "Free Software Definition" sets out four essential freedoms for free software (see text box above).[1] The FSF also maintains a list of approved free software licences that meet its criteria under the definition.[2]

The OSI adopts a more business-focused approach to FOSS, promoting the ways that enterprises and small businesses can benefit from using and releasing open source software. The OSI's "Open Source Definition" describes ten criteria for open source software (see table on the next page).[3] Where an open source licence meets these criteria, it becomes "OSI Approved".[4]

FSF: The Free Software Definition

A program is free software if the program's users have the following four essential freedoms:

- *The freedom to run the program, for any purpose (freedom 0).*
- *The freedom to study how the program works and change it so it does your computing as you wish (freedom 1). Access to the source code is a precondition for this.*
- *The freedom to redistribute copies so you can help your neighbor (freedom 2).*
- *The freedom to distribute copies of your modified versions to others (freedom 3). By doing this you can give the whole community a chance to benefit from your changes. Access to the source code is a precondition for this.*

[1] Free Software Foundation, "Free Software Definition" (2012), <http://www.gnu.org/>.
[2] Free Software Foundation, "Various Licenses and Comments about Them" (2012), <http://www.gnu.org/>.
[3] Open Source Initiative, "The Open Source Definition" (accessed 2012), <http://opensource.org/> [*OS Definition*].
[4] Open Source Initiative, "Open Source Licenses" (accessed 2012), <http://opensource.org/>.

OSI: OSS licences must permit and include:
1. Free Redistribution
2. Source Code
3. Derived Works
4. Integrity of Author's Source Code
5. No Discrimination Against Persons or Groups
6. No Discrimination Against Fields of Endeavor
7. Distribution of Licence
8. Licence Must Not Be Specific to a Product
9. Licence Must Not Restrict Other Software
10. Licence Must Be Technology-Neutral

2.2 Terms of FOSS Licences

Even amongst the licences that the FSF and OSI approve as sharing the basic characteristics of FOSS, the various terms and obligations vary greatly. However, there are several categories of clauses commonly encountered:

- General disclaimers of warranty and/or liability;
- Notice obligations, which generally require you to notify downstream users of the particular FOSS licence that applies to the work;
- Source code obligations, which generally require you to provide the source code of the software when you distribute it; and
- Hereditary licensing obligations, which generally require you to only release any modifications or improvements as FOSS, under the same licence.

The first two types of clauses – disclaimers and notice obligations – are present in nearly every FOSS licence. In fact, the short and popular 2-clause "FreeBSD licence" contains little more than these two bare obligations.[5] Other more expansive licences, such as the GNU General Public License (GPL), contain clauses setting out all four of these obligations.[6]

[5] FreeBSD Project, "FreeBSD Copyright" (1992), <http://www.freebsd.org/> [2-Clause BSD]; also see *infra* Appendix 3 at 56 (2-clause BSD licence summary).

[6] *Free Software Foundation*, "GNU General Public License Version 3" (2007), <http://www.gnu.org> [GPLv3]; also see *infra* Appendix 3 at 64 (GPLv3 licence summary).

2.2.1 Disclaimers

Most FOSS licences include a disclaimer of warranty that aims to be as exhaustive as possible in ensuring that no warranty applies. For example, the following clause in the Mozilla Public License Version 2 (MPLv2) demonstrates a typical incarnation of this licence term:

> *Covered Software is provided under this License on an "as is" basis, without warranty of any kind, either expressed, implied, or statutory, including, without limitation, warranties that the Covered Software is free of defects, merchantable, fit for a particular purpose or non-infringing. The entire risk as to the quality and performance of the Covered Software is with You. Should any Covered Software prove defective in any respect, You (not any Contributor) assume the cost of any necessary servicing, repair, or correction. This disclaimer of warranty constitutes an essential part of this License. No use of any Covered Software is authorized under this License except under this disclaimer.*[7]

As shown, in addition to setting out that the licence is "without warranty of any kind", these clauses are most often careful to exclude any warranty of "merchantability" or "fitness for a particular purpose". This serves to clarify and make certain that these two warranties do not apply, as some legislation might otherwise impose them as implicit terms of a contract or licence. For example, where software is sold in Ontario, the *Sale of Goods Act* imposes implied warranties relating to merchantability and, in some cases, warranties of fitness for a particular purpose.[8] However, with the explicit exclusions in typical FOSS licences, these warranties do not apply.

Many licences also explicitly disclaim warranties of non-infringement (for example, the MPLv2, as set out above). This disclaimer aims to exclude the U.S. doctrine of an implied warranty of non-infringement, which otherwise warrants that the software does not infringe any third party's copyright.[9] For example, if a software developer uploaded a piece of code to a FOSS project and it originally came from copyright-protected restricted-source software, anyone downloading or using the FOSS application would technically infringe the copyright. If sued for such an infringement, the warranty of non-infringement would result in such a user having no recourse against the distributors or developers – it is "users beware". In Canada, even though a specific doctrine of a warranty of non-infringement does not exist, this licence clause still likely

[7] Mozilla, "Mozilla Public License Version 2" (2011), <http://www.mozilla.org>, s. 6 [MPLv2]; also see *infra* Appendix 3 at 75. This primer presents clauses from the MPLv2 licence as examples throughout the paper because, released in 2012 following an extensive community consultation, the licence's legal language best reflects modern drafting practices of simplicity and plain language.

[8] *Sale of Goods Act*, RSO 1990, c S.1, s. 13 & 15 (s. 13 provides implied warranties related to merchantability such as warranting that the goods are free of encumbrances; s. 15 provides a warranty of fitness for a particular purpose in certain cases such as where the seller provides a description). Depending on the jurisdiction, consumer protection legislation and/or products liability legislation can also impose similar warranties.

[9] U.S., *Uniform Commercial Code*, § 2-312.

disclaims similar warranties of title (especially when considered in conjunction with a broad disclaimer against warranties of "any kind").[10]

As well as disclaiming warranties, nearly all FOSS licences disclaim liability. For example, the MPLv2 states:

> *Under no circumstances and under no legal theory, whether tort (including negligence), contract, or otherwise, shall any Contributor, or anyone who distributes Covered Software as permitted above, be liable to You for any direct, indirect, special, incidental, or consequential damages of any character including, without limitation, damages for lost profits, loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses, even if such party shall have been informed of the possibility of such damages. This limitation of liability shall not apply to liability for death or personal injury resulting from such party's negligence to the extent applicable law prohibits such limitation. Some jurisdictions do not allow the exclusion or limitation of incidental or consequential damages, so this exclusion and limitation may not apply to You.*

Again, this disclaimer broadly aims to cover all bases of liability and all types of damages that could occur. The result is that when you use FOSS, you do so at your own risk. The only exceptions are perhaps circumstances where notice of the disclaimer is not sufficiently brought to the user's attention, or the clause is unconscionable, in light of a particularly high-risk context.[11]

Although these disclaimers are standard in nearly all FOSS licences, licensors can modify them. In some cases, a FOSS licence allows the licensor to set out additional disclaimers. For example, the MPLv2 allows a licensor to "include additional disclaimers of warranty and limitations of liability specific to any jurisdiction."[12] In other cases, a licensor may do the opposite and choose to offer a warranty or accept liability, removing the effect of the disclaimer of warranty. For instance, the MPLv2 provides that "[y]ou may choose to offer and to charge a fee for warranty, support, indemnity or liability obligations to one or more recipients of Covered Software".[13]

---

[10] See e.g. *Sales of Goods Act*, *supra* note , s. 13(a).

[11] See e.g. *Tilden Rent-A-Car Co. v. Clendenning*, (1978), 83 DLR (3d) 400. Also note that click-wrap licences (e.g., "I Agree" buttons) and browse-wrap licences (e.g., hyperlinks to licences from web pages) are most often legally enforceable.

[12] MPLv2, *supra* note ,

[13] *Ibid.*

2.2.2 Notice Obligations

Like disclaimers, a notice obligation clause comes standard in nearly every FOSS licence. This is an obligation to reproduce the licence text (either directly or through a hyperlink) whenever you reproduce the licenced work. In addition, notice obligations usually require distributions of the work to retain other notices that come with it – for example, copyright assertions, additional disclaimers of warranty or liability, or patent notices.[14] Notice obligations usually at least apply to distribution of the source code, but, depending on the licence, often apply to redistribution of the object code as well.[15] As one example of a typical notice obligation, the MPLv2 licence provides:

> *You may not remove or alter the substance of any license notices (including copyright notices, patent notices, disclaimers of warranty, or limitations of liability) contained within the Source Code Form of the Covered Software, except that You may alter any license notices to the extent required to remedy known factual inaccuracies.*

2.2.3 Hereditary Licensing

There is one distinctive and commonly-encountered feature of FOSS that tends to divide licences into two categories: those with a **hereditary** obligation (also called "copyleft", "share-alike", or "reciprocal") and those without.[16] Licences in the hereditary category stipulate that modifications to the software must also come under the same licence. Therefore, if a person makes a change to hereditary software or includes some hereditary-licensed code in his or her own software, that person cannot distribute the new work as restricted-source software: he or she must only redistribute the new code under the same FOSS licence.

The GPL, which is the most popular copyleft licence, sets out the rationale for this stipulation as follows:

> *To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.*
>
> *For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code.*[17]

[14] *Ibid;* see also 2-Clause BSD, *supra* note .

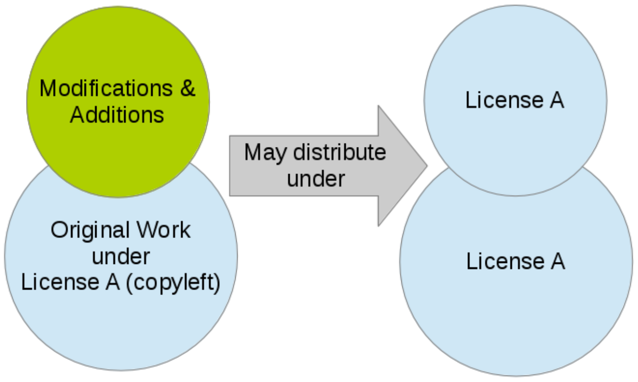
[15] See e.g. MPLv2, *ibid (*notice obligation applies to source code but not object code); 2-Clause BSD, *ibid.* (notice obligation applies to source code and object code).

[16] See e.g. Heather Meeker, *The Open Source Alternative* (Hoboken, NJ: John Wiley & Sons, 2008) at 22.

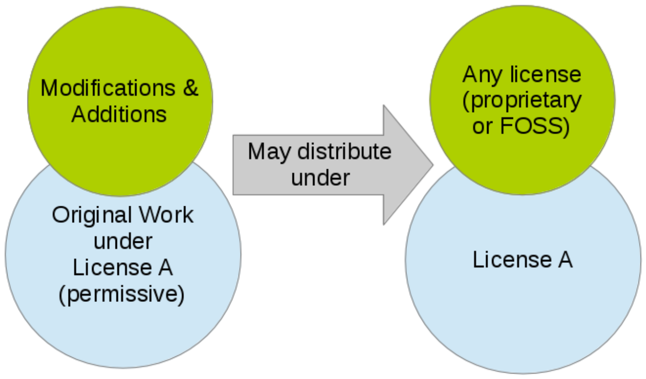
[17] GPLv3, *supra* note .

Only some FOSS licences contain this stipulation: other popular licences such as the BSD licence and the Apache licence do not. Licences without this stipulation are generally referred to as **permissive**. In these cases, the author of any modifications is under no obligation to place his or her changes under the same licence, or even under a FOSS licence at all. In general, the author of the modifications can even use and redistribute the modified software as a restricted-source software application. The two different types of FOSS licences are illustrated in the figure below.

A. Distribution under a hereditary licence


Modifications & Additions
Original Work under License A (copyleft)
May distribute under
License A
License A

B. Distribution under a permissive licence


Modifications & Additions
Original Work under License A (permissive)
May distribute under
Any license (proprietary or FOSS)
License A

However, blurring the distinction between these types of licences, different hereditary licences stipulate varying degrees of when this hereditary obligation to distribute under the same licence engages. This is perhaps the most confusing aspect of FOSS licences and deserves careful attention. To determine whether a certain activity implicates a hereditary obligation, one must consider (1) the *type of distribution* and (2) the *extent of integration* with the original code.

With respect to the type of distribution, hereditary obligations only arise upon certain "distribution"-like activities. Where there is no such distribution, the licences almost always allow a person to *use* and *modify* hereditary-licensed software without ever releasing their code. For example, a company can change and customize software under a hereditary licence for in-house use, and it does not need to share this software or the software source code. However, a "distribution" does trigger a hereditary obligation to distribute under the original licence. There are two common types of triggers found in FOSS licences:

- **Distribution of Source or Object Code**: Distribution of the software, either through the internet or on a physical medium such as a CD, whether as source code or object code, is the most common trigger for a hereditary obligation. For example, this is the trigger set out in the popular GPL licence.
- **Access over a computer network**: Found in "Affero" licences such as the "Affero GPL", access over a computer network is a much broader trigger that imposes a hereditary obligation whenever others access the licenced software over a computer network. This trigger aims to capture web service businesses that run their platforms on FOSS – these businesses must make their source code available to others.

Even if the software is distributed, the hereditary obligations still only extend to certain modifications (depending on the licence):

- **Derivative works:** As set out in the GPL, this type of hereditary clause stipulates that the obligation applies to all "derivative works" (the U.S. analog of an "adaptation" under Canadian copyright law, likely with similar scope). The exact boundary of what constitutes a "derivative work" is hotly debated; however, it is clear that a mere "collection" of works does not trigger hereditary obligations. A "collection" is where multiple software programs are distributed together (such as on the same CD-ROM), but where the programs do not tightly interact.
- **Derivative works w/a linking allowance:** The most prominent example of this type of clause is that found in the GNU Lessor General Public License (LGPL). This licence is similar to the GPL, but it explicitly allows a person to statically or dynamically link their code to a LGPL software library, without triggering the hereditary obligation (in software development, "linking" involves a loose coupling where one piece of the software communicates with other software and makes use of its functionality).
- **Modified files only:** Unique to the Mozilla Public License (MPL), this type of clause only imposes hereditary obligations on modified *files*. If a derivative work contains files which are modified versions form the original work, as well as entirely new files, only the directly-modified original files implicate and fall under the hereditary obligation.

The differences amongst the various licences in the popular GNU suite illustrate how the type of distribution and extent of integration interact to determine when a hereditary obligation engages, as shown in the following table.

Does the new work trigger the hereditary licensing obligation?			
	Derivative work of the original	**Derivative work, but only links to original**	**Collection, including the unchanged original**
Distribution of source code or object code	GPLv3[18]: **Yes** LGPLv3[19]: **Yes** AGPLv3[20]: **Yes**	GPLv3: **Yes** LGPLv3: **No** AGPLv3: **Yes**	GPLv3: **No** LGPLv3: **No** AGPLv: **No**
Provision of access over a computer network	GPLv3: **No** LGPLv3: **No** AGPLv3: **Yes**	GPLv3: **No** LGPLv3: **No** AGPLv3: **Yes**	GPLv3: **No** LGPLv3: **No** AGPLv3: **No**

[18] Free Software Foundation, "GNU General Public Lessor Version 3" (2007), <http://www.gnu.org> [GPLv3]

[19] Free Software Foundation, "GNU Lesser General Public Lessor Version 3" (2007), <http://www.gnu.org> [LGPLv3]

[20] Free Software Foundation, "GNU Affero General Public Lessor Version 3" (2007), <http://www.gnu.org> [LGPLv3]

The subsistence and extent of the copyleft feature is generally the FOSS issue of paramount importance to FOSS users and contributors. Where a business relies upon revenues from the licensing and sale of restricted-source software products, the use of hereditary software in its code base could ultimately attach the hereditary obligation to the whole of its software product, creating an unwanted obligation to release the source code for the entire product.

2.2.4 Source Code Obligations

Hereditary licences usually include a "source code obligation" that requires anyone distributing the work to include, or at least link to, the source code. This ensures that the software remains in a format that downstream users can continue to modify. Without such a requirement, someone could make changes to software under a hereditary licence, but only redistribute the object code. This would make it impossible for others to make further changes or re-integrate any of the changes back into the original project.

Although a source code obligation usually goes hand-in-hand with a hereditary licence, it is worth noting that, in some cases, it may only attach to the originally-licensed code. This becomes particularly important when using software libraries. For example, restricted-sourced software vendors often link-to and make use of FOSS libraries that fall under a Lesser GNU Public License.[21] As long as the restricted-source software merely links to these libraries, the hereditary obligation does not engage. The vendor has no obligation to release its own source code. However, the source code obligation still applies to the libraries themselves. The vendor has an obligation to redistribute or otherwise provide the *original* library source code along with the restricted-source product.[22]

In some cases, it is not feasible for a person to directly provide the source code along with the object code. If distributing software on a CD or DVD, space constraints may make it difficult to include both – especially considering the fact that source code can be much larger than object code. For this reason, licences are usually flexible about the manner in which you must provide the source code. For instance, the GPLv3 allows a person to provide, in lieu of the source code itself, a written offer to distribute the source code on a separate CD or through a network server.[23] If distributing the object code over the internet, a person may also simply provide instructions on how to download the source code from another internet-connected server.

[21] LGPL, *supra* note.
[22] *Ibid.*, s. 4(d).
[23] GPLv3, *supra* note 18, s. 6(b).

3. Use Cases & Applicability of FOSS Licence Obligations

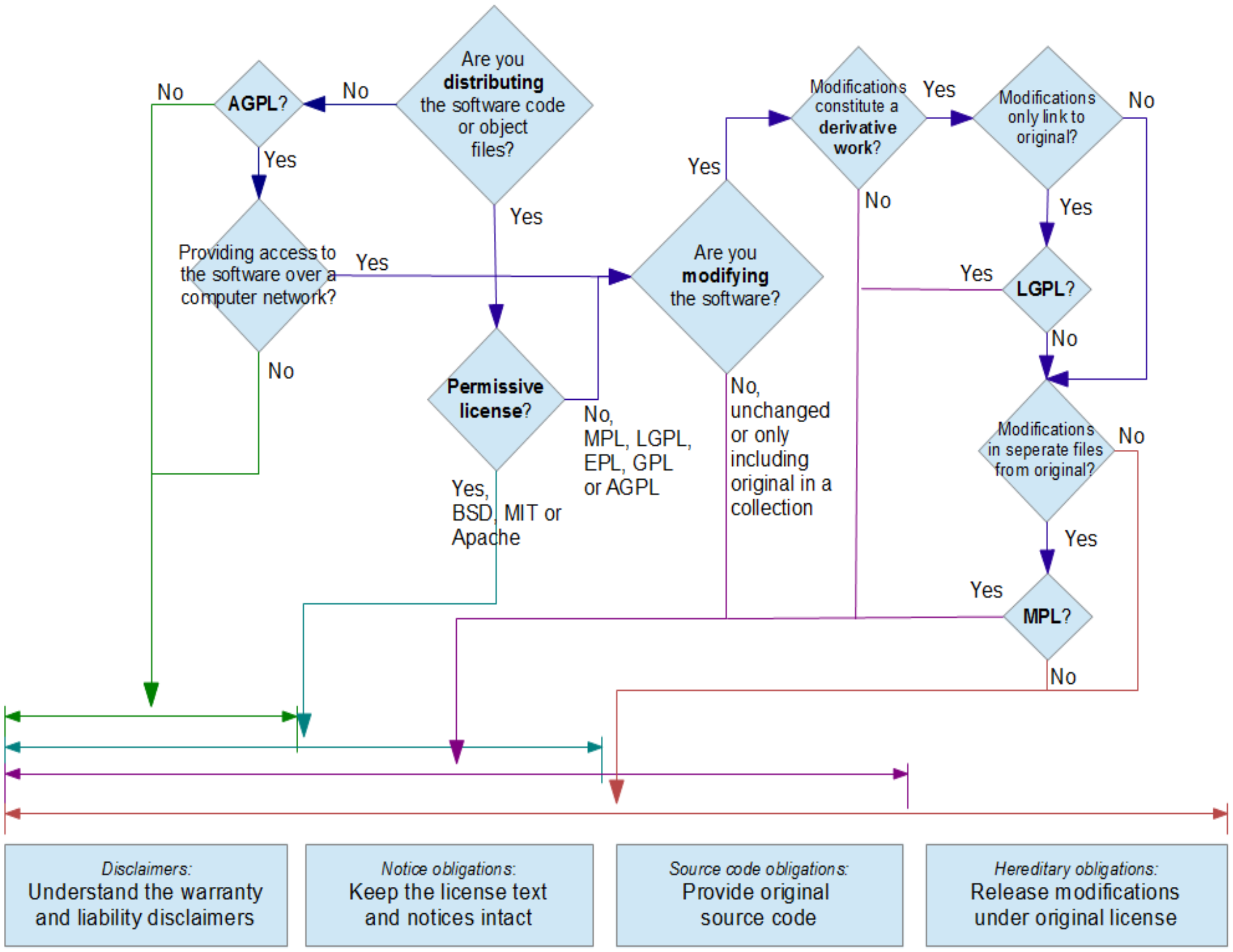
3.1 General Use Cases and Corresponding Obligations

There are four typical use cases of FOSS: one associated with mere use of FOSS and the other three associated with various cases of distributing FOSS. Each of the four uses cases distinctively map to different sets of licensing obligations:

- Use Case A: Use without any distribution, engaging:
  - acceptance of the disclaimer of warranty and liability.
- Use Case B: Distribution without any modification, engaging:
  - the above;
  - notice obligations; and
  - any source code obligations;
- Use Case C: Distribution with modifications, engaging:
  - all of the above; and
  - any hereditary licensing obligations.
- Use Case D: Distribution of new software, engaging:
  - *no obligations* – when it is your software, you may use, distribute and license it as you choose.

3.2 Licence-Specific Obligation Map

Of course, this general mapping is only a guideline. Nuances in particular licences often alter these sets of obligations. The following decision flowchart more fully sets out these core obligations for the most popular FOSS licences:[24]


Are you distributing the software code or object files?
No
AGPL?
No
Yes
Providing access to the software over a computer network?
Yes
No
Yes
Permissive license?
No, MPL, LGPL, EPL, GPL or AGPL
Yes, BSD, MIT or Apache
Are you modifying the software?
Yes
No, unchanged or only including original in a collection
Modifications constitute a derivative work?
Yes
No
Modifications only link to original?
No
Yes
LGPL?
Yes
No
Modifications in seperate files from original?
No
Yes
MPL?
Yes
No
Disclaimers: Understand the warranty and liability disclaimers
Notice obligations: Keep the license text and notices intact
Source code obligations: Provide original source code
Hereditary obligations: Release modifications under original license

[24] Chart sets out the core obligations only for the BSD licenses (2-clause and 3-clause), MIT license, GPL license suite (versions 2.0 to 3.0), Apache License 2.0, Eclipse Public License 1.0 (EPL), and the Mozilla Public License (MPL, versions 1.1 & 2.0).

4. Use of FOSS

4.1 Typical Scenarios of FOSS Use (Use Case A)

Within the context of FOSS use where one does not distribute the software, there are five scenarios in which organizations typically make use of FOSS: web and file services, desktop office suites, specialty software applications, customized software for in-house use, and customized software for use within other departments in the same organization.

4.1.1 Server-side software

Especially on the internet, FOSS has long been a cornerstone of server software. FOSS web server software such as Apache and Nginx run more than 70% of the websites on the internet.[25] The FOSS database server MySQL is widely popular, and the FOSS-licensed TomCat software captures a majority of the application servers market.[26] Additionally, the Linux operating system frequently runs the back-end of these servers.[27]

Public sector organizations deploy FOSS for servers as readily as private sector organizations. For example, the primary Government of Canada website at www.canada.gc.ca runs on the open source Apache web server.[28] The National Research Council uses Apache, MySQL, and the OpenLDAP authentication tool for electronic publications.[29] In fact, overall, server-side software remains the primary use of FOSS within public sector organizations.[30]

4.1.2 Desktop Productivity Software

Compared to server-side FOSS use, institutional and business use of open source desktop software is a relatively new occurrence.[31] However, this area of open source software has seen increased interest in recent years, especially in the public sector.[32] For example, public

[25] See Netcraft, "August 2012 Web Survery" (2 August 2012), <http://news.netcraft.com/> (Apache and Ngix collectively capture 70.92% of the market across all domains).

[26] Jelastic, "Software Stack Market Share: July 2012" (23 July 2012), <http://blog.jelastic.com/>

[27] *Ibid*, s. 6(d).

[28] Treasury Board of Canada Secretariat, "Open Source Software Frequently Asked Questions" (2007), <http://www.collectionscanada.gc.ca/>.

[29] *Ibid.*

[30] Kris Ven, Dieter Van Nuffel & Jan Verelst, "The Migration of Public Administrations Towards Open Source Desktop Software: Recommendations from Research and Validation through a Case Study" in Sulayman Sowe, Ioannis Stamelos & Ioannis Samoladas, eds., *Emerging Free and Open Source Software Practices* (Hershey: IGI Publishing, 2008) at 192-193.

[31] *Ibid.*

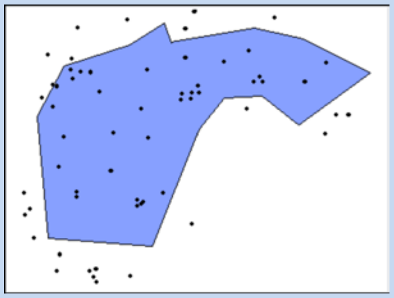
[32] *Ibid.*

administrators in the City of Paris decided to migrate 17,000 desktop computers with Microsoft Office and Microsoft Internet Explorer over to using OpenOffice.org and Mozilla Firefox.[33]. Public administrators in the City of Munich have migrated 14,000 desktop computers to the Linux operating system.[34]

Although Libre/Open Office and Firefox are amongst the most common FOSS desktop productivity applications, a broad range of applications are available. The Open Source Software Directory lists over one thousand FOSS alternatives to proprietary software applications.[35]

Example - Use of FOSS "Off-the-Shelf": What obligations apply?

Project Manager: "Dave, I am emailing you a list of fifty geodetic points (latitude and longitude for each point). We need to know whether each of these points lies within city boundaries."



Task Resolution: Dave, a software developer, finds a function called *inpolygon* in the software package GNU Octave that can solve this problem (GNU Octave is similar in functionality to Matlab). He downloads the software, installs it on his machine (*apt-get install octave)*, then runs it and checks each geodetic point.

When he launches the program, Dave also sees a notice asking him to type "warranty" for further information on the licence. He does so and the finds that GNU Octave is licensed under the GNU General Public License v3 (GPLv3).

What legal terms apply?

- ✓ **Disclaimers of warranty and liability?** Yes, as set out in s. 15 & 16 of the GPLv3.
- ✗ **Notice obligations?** No, the notice obligations under GPLv3 only apply when you "convey" a copy.
- ✗ **Hereditary obligations?** No, the hereditary obligations under GPLv3 only apply when you "convey" a copy, and only to modified versions.
- ✗ **Source code obligations?** No, the source code obligations under GPLv3 only apply when you "convey" a copy.

[33] *Ibid.* at 195.
[34] *Ibid.*
[35] Open Source Software Directory, "About" (accessed August 2012), <http://www.opensourcesoftwaredirectory.com> (reports a listing of "1022 applications").

4.1.3 Specialty Software

Many FOSS utilities and applications serve a particular business area or other interest. In the case of highly-specialized software, alternatives to a particular FOSS application may not even exist if no sufficiently large market has developed to attract restricted-source vendors. In many cases, academic researchers directly release their specialized research initiatives as FOSS software libraries and applications.

As one example within the specialty area of geomatics, MapServer is a FOSS package commonly used to display dynamic spatial maps over the internet. MapServer was originally developed at the University of Minnesota.[36] Released under FOSS licence, it is now administered by the Open Source Geospatial Foundation (OSGeo) and used and supported by a worldwide developer community.[37]

4.1.4 Internal Modification

All FOSS licences grant users broad rights to modify the software.[38] Recalling that hereditary obligations only engage upon distribution, one can exercise this right to modify the software for internal use without triggering any additional legal obligations. The legal context of modifying FOSS and using it internally is the same as merely using it internally without modification.

However, although there is no legal difference between these two scenarios, some organizations and businesses may still wish to establish a policy of handling modified software differently.[39] Tracking and handling modified software with extra caution can help ensure that an organization or business does not mistakenly distribute the software at a future time (which would then raise additional legal implications).

4.1.5 Internal Distribution

The scenario of "use without distribution" (i.e., the context which does not engage hereditary licensing obligations) is relatively broad in scope. In general, internal distribution within a business or organization does not constitute a legal "distribution" or "conveyance" and does not implicate hereditary licensing obligations. For example, the GPL considers distribution as "any kind of propagation that enables *other parties* to make or receive copies" [emphasis added]. A corporation is only a single legal party and can make, modify and distribute FOSS to its employees without engaging additional obligations.

[36] Regents of the University of Minnesota, "About" (2011), <http://mapserver.org/>.
[37] *Ibid.*
[38] See Free Software Definition, supra note 2, freedom #1.
[39] See e.g. Canada, National Research Council, "NRC Open Source Software Guidelines" (2012), Annex A.

Given that separate government departments operate semi-autonomously, there is a possible argument that the different departments constitute different "parties". However, this interpretation is unlikely. Such an understanding would be incongruent with the general legal understanding of a federal or provincial Crown as being a single legal entity. Even the Free Software Foundation, which typically promotes a strong and wide understanding of distribution, takes the view that "[f]ederal Government agencies may share free software without making a 'distribution'".[40]

However, distribution between the different legal entities of the federal crown and a provincial crown, or between provincial crowns, almost certainly constitutes a "distribution".

4.2 Policies and Perspectives on FOSS Use

A key feature of FOSS is that anyone can freely distribute the software. As a result, most FOSS is available for download on the internet without cost, but there are certainly still maintenance and management expenses. For businesses choosing to use FOSS, the nil up-front licensing cost is an attractive and key policy driver for making this decision. It can lower operating expenses. For the public sector, cost is also important. In addition, other public benefits can motivate public sector use of FOSS. Of course, a business or government organization must balance these benefits against several drawbacks, such as lower user familiarity with FOSS software packages.

4.2.1 Benefits

Cost

The total cost of ownership for using any software application involves three over-arching categories of expenses: (1) up-front licensing costs, (2) on-going maintenance and support costs, and (3) software upgrade costs.

When using FOSS, the up-front licensing cost is zero (aside from internal costs of assessing the software and the licence). A FOSS licence is always "open" and granted to the world, such that it licenses everyone to use the software without any fees due. Moreover, due to the fact that FOSS licences grant everyone the right to redistribute the software, nearly all FOSS packages are available for free download over the internet. At most, some companies charge a nominal fee for the service of distributing FOSS on a CD or DVD.[41]

[40] Letter from Eben Moglen to Brian Fitzgerald (3 December 2003), cited in Brian Fitzgerald and Nic Suzor, "Legal Issues for the Use of Free and Open Source Software in Government", (2005) Open Source Software 412 at 435.

[41] E.g., Canonical sells CDs of the Linux distribution Ubuntu for £5 (U.K.) (Canonical, "CDs and DVDs" (accessed 2012), <http://canonical.com>) [*Ubuntu CDs*].

As well, FOSS software often incurs lower on-going maintenance costs. With restricted-source software, the vendor and its select business partners are quite often the only businesses able to provide adequate support (either because the software licence grants the vendor an exclusive support contract, or due to the vendor's specialized expertise with the software and its sole ability to examine and work with the source code). This can potentially undermine competitive tendering and, in some cases, result in poor support with no available alternative.[42] FOSS, on the other hand, permits anyone to install, fix, and otherwise support the software. This can allow for a more competitive tendering of support services amongst firms.

With respect to software upgrades, some restricted-source vendors require licensees to purchase a new licence, or pay an upgrade fee, for new versions of the software. For FOSS, on the other hand, no new licence is necessary to start using a new version of the software.

Security

When using FOSS, all of the source code is openly published. This makes it difficult for anyone to surreptitiously insert malicious code. It also allows security auditors and security researchers to inspect the code for flaws.

While software security is a concern for all entities, it is of paramount importance to governments. For this reason, defense and national security agencies very often use FOSS.[43] Fadi Deek and James McHugh explain the security benefits:

> *A key factor in the attractiveness of open software in security (and national security) applications is its auditability. It is obviously harder to conceal things in open source code, while, conversely, governments may have reason to be leery of what may lurk inside proprietary code...In the U.S. context, the major proprietary vendor Microsoft is a domestic corporation, so at least the government can be expected to work out disclosure mechanisms with the vendor. However, this is a less likely scenario for foreign-held entities. For example, is Microsoft likely to disclose proprietary code to the government of Venezuela because that government wants to scrutinize Microsoft applications for security flaws or traps?*[44]

Thus, where FOSS is well-developed and well-inspected by third parties, security is generally improved. There are some security risks as well, discussed below.

[42] Roger Petry, "Free Software: Ideal for Saskatchewan" (2003) 3:5 CCPA Saskatchewan Notes 1, <http://www.policyalternatives.ca>.

[43] Fadi Deek and James McHugh, "Open Source Technology and Policy" (New York: Cambridge University Press, 2008) at 310-311 (extensive use of FOSS within U.S. D.o.D. and NSA) [*Open Source Technology and Policy*].

[44] *Ibid.*

Avoiding Vendor Lock-in

When a restricted-source software application becomes entrenched into an organization's or business's processes or products, the organization becomes beholden, or "locked-in", to that software vendor for any new features, for bug fixes, and in many cases, for receiving product support. Where a vendor is unwilling to implement a new feature, you may need to switch to an alternative – often at considerable cost. Where a particular vendor is slow to provide bug fixes or support, this can adversely affect your own timelines and may also impose a security risk.

FOSS provides an advantage in that it creates an open marketplace for providers of all types of support. Any support business with sufficient software development competencies can add new features and fix bugs in the software; FOSS users can also switch to a different support provider whenever an existing company no longer meets their needs or timelines.

Support for the Local Economy

Depending on the dynamics of a software industry in a particular locality, FOSS use may better support local businesses in the area.[45] Because FOSS is openly available, distributable, and modifiable, a wider breadth of smaller "support service" businesses may provide commercial services. Rather than a single vendor providing the warranty and technical support, any competent local IT business can provide these services. Rather than a single vendor being the only party in a position to customize software, a local consulting or software development business can provide specialized versions of the software.

4.2.2 Risks and Drawbacks

Lower User Familiarity

User familiarity with FOSS software packages is often much lower than with restricted-source software packages. Restricted-source software has a strong foothold in schools (at least in Canada) and many people learn to use computers through restricted-source software, due to its pre-installation by hardware suppliers. For this reason, FOSS can require more training and support.[46]

[45] *Ibid.*

[46] See generally Migration of Public Administrations, supra note 30.

Security Risks

Because all of the source code of FOSS is openly published, anyone – including "black hat hackers" – can examine the code for security holes.[47] In particular, malicious attackers can observe where FOSS software shares the same design, code base, or architecture as software that may have known security vulnerabilities.[48]

Of course, one must weigh this drawback against the aforementioned security benefits. In many cases, malicious hackers can still obtain access to the source code of restricted-source software where non-disclosure agreements, company ethics procedures, or vendor security mechanisms fail. Whereas restricted-source software security depends on an attempt to maintain an information imbalance between the "white hats" (computer security analysts) and the "black hats" (malicious computer hackers) , FOSS software security depends upon an open competitive process of finding and closing security holes. The models rely on different strategies, and each has its particular risks that an organization must weigh and consider.

Loss of Control over Software Management

The ease and no-cost option of downloading FOSS from the internet, and instantly installing it, can create an attractive proposition for employees to side-step procurement processes altogether. This creates two problems: first, employees may contractually commit the company or government organization to the terms of the software licence without management even becoming aware of this fact; and second, losing track of what software is running on employee workstations makes systems security management more difficult. However, this systems management issue subsists whether or not a company or organization chooses to make use of FOSS.

[47] Kirk St. Amant & Brian Still, Handbook of Research on Open Source Software (Hershey: Information Science Reference, 2007) [Handbook on OSS] at 210.

[48] *Ibid*. at 192

4.3 Best Practices for FOSS Use

4.3.1 Software Evaluation and Procurement

Evaluating Software Features

In general, the same factors applicable to an evaluation of the feature set and maturity of restricted-source software also apply to FOSS. Based on a literature survey, Kirk St. Amant and Brian Still identify nine factors that merit consideration when evaluating FOSS, as shown in the following table.[49]

Factors	Description
1. Community	A strong user community involved in a project provides people to answer questions, test the software, report bugs, suggest improvements and drive forward overall interest in the software.
2. Release Activity	A strong developer community with a history of releases and continued involvement tends to demonstrate that fixes and improvements to the software will continue into the future.
3. Longevity	Longevity, measured in both in terms of the age of the product and the number of versions released, indicates a project's stability and chance of survival.
4. Licence	The licence of a project can affect the level of legal risk which you must assume (see “Legal Risk Management” below).
5. Support Availability	Support considerations include user support (i.e., the availability of assistance with installation and usage) and maintenance (i.e., fixing problems in the software). Support for FOSS can be provided by the community and/or paid support services businesses.
6. Documentation	User documentation provides important information to help users install software and use its features.
7. Security	Although FOSS code is auditable, this does not necessarily mean it is secure. The quality of the code and the typical response time for patching security-related flaws help indicate the security level of the software.
8. Functionality	Specific functionality needs depend on the business case for the software and need to be assessed on a case-by-case basis.
9. Integration	Where a software package will interact with other pieces of software, or with particular data formats, compatibility and the ability to integrate the software and data together becomes a paramount consideration.

[49] *Ibid* at 197-210.

The free OHLOH tool (http://www.ohloh.net) provides many useful metrics to help with evaluation of FOSS. For example, it lists the age of the project, the number of contributors, and the frequency of software updates.[50]

Comparing FOSS and Restricted-source software

In making software procurement decisions, it is common practice to assess FOSS on equal footing with restricted-source software.[51] Starting from a list of business requirements, a government organization or business can assess how well the feature set of a FOSS software package matches the requirements, and calculate the overall cost-per-feature over the required lifetime of the software.

This type of assessment provides a useful starting point. However, several interviewees reported difficult experiences in making “apples-to-apples” comparisons between FOSS and restricted-source software. Although an assessment of the total cost of ownership is feasible for both, it is difficult to account for factors such as benefits to the public (e.g., overall economic efficiency, support for local businesses, etc.).

Compounding this difficulty, established software procurement and service tendering processes are sometimes tailored towards restricted-source solutions. The processes are not always flexible enough to properly assess and address FOSS solutions. For example, a procurement process may first involve a call for tenders on the initial licensing contract. Here, FOSS-based businesses are not likely to bid, as they have no licences to sell.[52] Where the next stage of a procurement process would call for tenders on support services, FOSS-based support businesses may already be closed out. Restricted-source software vendors often prohibit anyone else from servicing or providing support for their product, thereby barring FOSS-based businesses from the whole process.[53]

Thus, although a direct comparison between all potential solutions – both FOSS and restricted-source – is a helpful practice, it may also be necessary for businesses and organizations to revisit existing processes and procedures in order to ensure that these adequately envision both categories of software.

[50] Black Duck Software, “Ohloh” (accessed September 20120), <http://www.ohloh.net/>.
[51] See e.g. BC Public Sector, “IM/IT Enablers Strategy v1.5” (2011), <http://www.cio.gov.bc.ca> (Government of BC policy).
[52] Emily Chung, “An Open Door for Open Source?”, *CBC News* (12 February 2009), <http://www.cbc.ca>.
[53] *Ibid.*

4.3.2 Training and Support

Depending on the context, it is prudent to allocate appropriate resources for training and assistance with FOSS. Although this may not be necessary when merely installing new FOSS on a server (where only knowledgeable staff are likely to interact with it), it is particularly important when migrating over to FOSS desktop software.[54] Often, users are more accustomed to traditionally-popular restricted-source software packages, and may have little familiarity with FOSS. Organizations which have made the switch to FOSS desktop applications identify several helpful lessons learned and best practices, as indicated in the following table.[55]

Category	Lessons Learned and Best Practices
Analysis and Preparation	• Analyze and plan the costs and business cases for the migration • Conduct pilot studies
Migration	• Moderate the pace of migration, avoiding a "big bang" approach (e.g., first institute voluntary adoption to cultivate a base of experienced users who can then assist their colleagues) • Provide a transition phase where users have access to both existing software and the open source alternative • Secure the support of top management • Create a positive attitude with users
Training	• Provide proper training and focus on the general everyday functionalities of the software
Support	• Provide access to several kinds of support, such as FAQs and guides, access to telephone and email support, and access to a contact person
Format conversion	• Make a significant effort to ensure that file and data formats from the incumbent restricted-source software can be easily imported and exported by the FOSS solution • Keep in mind that document conversion, such as between MS Office and Open Office, can become a labour-intensive task for any complex documents, especially where they include precise formatting or MS Office macros
Functionality	• Although the general functionalities are often equivalent in the FOSS analogues, there is not always a direct alignment of feature sets and specific issues may arise for users

[54] Kris Ven, Dieter Van Nuffel & Jan Verelst, "The Migration of Public Administrations Towards Open Source Desktop Software: Recommendations from Research and Validation through a Case Study" in Sulayman Sowe, Ionnis Stamelos & Ioannis Samoladas, eds., *Emerging Free and Open Source Software Practices* (Hershey: IGI Publishing, 2008) at 193.

[55] See *ibid.* at 197-198.

4.3.3 Legal Risk Management

Overall, the particular FOSS licence is unlikely to strongly impact a decision on whether or not to use FOSS. As previously noted, few licence obligations engage upon the mere use of FOSS. Most importantly, hereditary licensing obligations do not apply. Only three features of FOSS licences tend to strongly differ from restricted-source licences in this context and merit special consideration: (1) the lack of a warranty; (2) the disclaimer of liability and lack of indemnification; and (3) the lack of choice of law and choice of forum clauses.

Lack of a Warranty

A warranty is an assurance by a vendor that a product will operate as promised and without any defect. A warranty can also include assurances that a product is free of legal encumbrances, such as unlicensed intellectual property that may be owned by third parties (i.e., a warranty of non-infringement). Depending on the terms of the warranty, when a product does not fulfill the assurances, the customer can request that the vendor fix the product, refund the customer and/or provide monetary compensation.

Given the typical lack of a warranty in a FOSS licence, it becomes more imperative for a business or government organization to secure external support contracts. A FOSS support services business familiar with the software can help resolve any issues and, importantly, directly patch any bugs that crop up.[56]

Some support services businesses also help mitigate the risks that can arise due to the lack of a warranty of non-infringement. For example, Red Hat offers customers the "Red Hat Open Source Assurance Program" and Canonical provides the "Ubuntu Advantage Assurance".[57] Both of these programs provide services to replace any portion of their respective Linux distributions (Red Hat and Ubuntu) that turn out to infringe intellectual property rights of other parties. Additionally, these programs also offer to indemnify customers against any lawsuits they face as a result of any such intellectual property infringements.[58]

Disclaimer of Liability and Lack of Indemnification

Where a licence or contract contains no disclaimer, a licensee can ordinarily hold the licensor civilly liable for losses suffered due to any negligence in the software design, implementation or testing. Further, where a licensor agrees to indemnify the licensee, if another party (most often, a customer of the licensee) sues the licensee for damages and the cause of the damages can be

[56] E.g. Canonical, "Ubuntu Advantage Support" (accessed September 2012), <http://www.canonical.com/> (Canonical provides support for the Ubuntu Linux distribution); Red Hat, "The Value of Your Red Had Subscription" (2010), <http://www.redhat.com> (Red Hat provides supports for the Red Hat Linux distribution).

[57] Canonical, "Ubuntu Advantage Assurance" (accessed September 2012), <http://www.canonical.com/> (Canonical provides support for the Ubuntu Linux distribution); Red Hat, "Open Source Assurance FAQs" (accessed September 2012), <http://www.redhat.com>.

[58] *Ibid.*

traced back to the original licensor's software, then it is the licensor who must ultimately pay a damage award.

However, given the disclaimer of liability and lack of indemnification clause in a FOSS licence, organizations and businesses cannot shift or "flow through" their civil liability risks to the vendor. The disclaimer means that you cannot generally recover compensation from the vendor for damages that arise due to problems with the software. The lack of indemnification means that, if other parties suffer losses due to problems involving your use of the software, you risk liability and cannot shift this risk onto the original software vendor.

Overall, the indemnity clauses in restricted-source software licences function similarly to insurance: they cover your losses when third parties sue you. Therefore, when it comes to FOSS, companies and organizations may wish to consider insurance as a substitute for an indemnity clause. Instead of the software vendor insuring against legal liability, an insurance provider does the same.

No Choice of Law or Choice of Forum Clause

One other difference that FOSS users should keep in mind is that FOSS licences rarely include choice of law or choice of forum clauses. Restricted-source licences may specify that courts shall interpret the licence and resolve disputes pursuant to the laws of a particular country or jurisdiction (choice of law) and that lawsuits shall be heard by courts in a particular jurisdiction (choice of forum). Where these remain unspecified in FOSS licences, courts will apply standard conflict of law rules to determine the appropriate law and the appropriate forum in a particular context. This increases the uncertainty of whether you may have to litigate a lawsuit in a foreign jurisdiction, or under unfamiliar laws – which would involve higher litigation expenses.

4.3.4 Community Participation

FOSS is built upon, and depends upon, collaboration and community participation. It is generally a good practice to try to "give back" to the community from which you receive benefits. Not only does such reciprocity help keep a FOSS software project alive and well, but it also helps you establish ties and a good rapport with other community members. This, in turn, can help when you request community support or submit feature requests.[59]

[59] See Julie Wang and Dan Robey, "Social Capital in OSS Communities: A Cross-Level Research Model" in Hind Benbya and Nassim Belbaly, eds., *Successful OSS Project Design and Implementation* (Surrey, England: Gower Publishing Limited, 2011) at 87ff.

Even without contributing any code, there are numerous other ways that you can contribute to a FOSS software project. For example, any user might consider one or more of the following activities:

- *Submit software suggestions*: provide bug reports and suggest any new features that could improve the software;
- *Improve documentation*: help write new user guides, correct and improve existing documentation, or submit artwork such icons, backgrounds and logos;
- *Assist others*: participate in help forums and product support mailing lists;
- *Make financial contributions*: especially, if a user is profiting from the software , the user may to share some it of it back to the community through donations to the project;
- *Contribute to the infrastructure*: contribute hardware, donate server time, or even help maintain content on the project web portal;
- *Promote the software*: espouse the benefits of the software to others and write reviews.[60]

4.4 Interoperability and Licence Alignment

Licence interoperability is rarely an issue for mere FOSS use, as long as there is no distribution of the software. As previously discussed, generally the only licence obligations that one must accept and follow in the case of mere use are a disclaimer of warranty and disclaimer of liability. Even when software from multiple sources is modified and combined under multiple licences, these disclaimers are never likely to conflict. Disclaimers impose only a passive obligation – that is, they do not require you to take any particular action or forebear from any particular action. A licensee need only accept the legal risk imposed.

Although *licence* interoperability is not a concern for FOSS use, *format* and *data* interoperability can affect decisions on whether or not to use FOSS. To assess the interoperability of data, it is important to conduct a case-by-case analysis. Restricted-source software can still implement widely-interoperable "open standards" for data. By the same token, FOSS may in some cases use specialized non-published and non-open standards for storing data.

However, cultural tendencies and the dynamics of the FOSS business model tend to drive FOSS projects towards the use of interoperable and open standards for reading, writing and storing data.[61] Indeed, even where FOSS does not implement an open standard for data interchange, at the very least the availability of the source code always provides an "open implementation". The source code serves as an example that anyone else can follow to implement the same functionality for reading and writing the underlying data format.

[60] Black Duck Software, "How to Go Open: Successfully Open Sourcing Internal Software" (Legal Webinar Series, 2011)

[61] See Katherine Noyes, "10 Reasons Open Source Is Good for Business", *PCWorld* (5 November 20120), <http:www.pcworld.com>.

Where governments use interoperable data formats compatible with FOSS applications, this can help enhance the governments' ability to communicate with citizens.[62] Anyone with internet access can download a FOSS application to open a compatible document, spreadsheet or other file; in contrast, restricted formats without FOSS support might require citizens to purchase additional software, creating an accessibility barrier for some.

In the geospatial systems domain, international collaboration is commonplace. Thus, in this context, it is even more imperative to ensure that colleagues in other nations, and those with fewer financial resources (such as colleagues based at academic or public institutions facing austerity budgets), can access and contribute to the inputs and outputs. Before choosing a software application, businesses and organizations should weigh whether the overall cost of a software application – whether restricted-source or FOSS – could be prohibitive to those working in other countries and other environments.

[62] Migration of Public Administrations, *supra* note at 193.

5. FOSS Distribution

5.1 Typical Scenarios of FOSS Distribution without Modification (Use Case B)

Where a business or organization redistributes FOSS as-is, without making any changes to it and without combining it with other software, this often still engages an additional legal obligation to redistribute the original source code. It does not typically engage any hereditary licensing obligations (refer to chapter 3, *supra*, for further details). Two common scenarios that fall under this use case are when a vendor sells FOSS, and when a support service provider supplies FOSS to a customer.

Example – Distribution of Original FOSS: What obligations apply?

Project Manager: "Dave, our clients require the ability to check for themselves whether geodetic points lie within city boundaries. Can you please package the GNU Octave software (the binaries that our clients can directly run) along with our city boundary shape files, and then email this software to them?"

What legal terms apply? Dave looks over the GPLv3 license that applies to GNU Octave and finds that the following apply:

- ✓ **Disclaimers of warranty and liability?** Yes, as set out in s. 15 & 16 of the GPLv3.
- ✓ **Notice obligations?** Yes, as set out in s. 4 of the GPLv3 ("Conveying Verbatim Copies").
- ✗ **Hereditary obligations?** No, the hereditary obligations under GPLv3 only apply to modified versions. Dave merely needs to package additional shape files with the original software, as a "collection" of the two.
- ✓ **Source code obligations?** Yes, as set out in s. 6 of the GPLv3 ("Conveying Non-Source Forms").

Legal compliance: To comply with the source code obligations, Dave considers his various options under s.6 of the GPLv3 and decides to "convey the object code by offering access from a designated place… (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge" (s. 6(d)). He puts his GNU Octave package in a private FTP directory, places the source code in the same directory, then gives the clients access to both.

To comply with the notice obligations, Dave looks through the software source files and finds (a) a file named README with a copyright notice ("Copyright (C) 1996-2012 John W. Eaton") and a disclaimer of warranty; (b) a file named COPYING setting out the GPLv3 licence terms; and (3) further disclaimers and copyright notices at the top of each source file.

Dave leaves all of these notices intact without removing or changing them.

5.1.1 Sale of FOSS

Recall that "free software" refers to the freedoms granted, not the price of the software. Although the freedom to redistribute FOSS creates a market where the price tends towards zero, vendors may still choose to sell FOSS, usually for a marginal fee.

For example, Canonical sells CDs and DVDs of its Linux distribution, Ubuntu, for a few dollars.[63] Customers may find it convenient to purchase FOSS on a physical medium for use in environments not connected to the internet, or where internet access is too slow to download a large software package such as a Linux distribution.

5.1.2 Support Service Provider Installation of FOSS

Where an organization or business externally contracts IT services and requires a FOSS application, the IT contractor may download the software herself or himself, then perform the installation – which most likely constitutes a distribution to a different party and thereby implicates the additional obligations that arise when distributing FOSS.

5.2 Typical Scenarios of FOSS Distribution with Modification (Use Case C)

One of the major benefits of FOSS is that all users of the software are free to modify it. This differs greatly from the restricted-source software context, where typically only the original company that developed the software can make any changes. With FOSS, the right to modify the software allows other developers to add new features, customize the software, and develop other software projects that build upon and use FOSS.

5.2.1 Addition of New Features and Customized Versions

A business or government organization may find that a particular FOSS application meets most of its business needs, minus only a few requirements or necessary customizations. In this case, the business or organization can simply add these features. In fact, the direct need for a particular feature not available from other sources is a primary driver for participation in FOSS.[64] To achieve the requisite functionality, a developer need not start from scratch, but rather can adapt an existing project to his or her needs.

Where a business or government organization has the capacity in-house to develop new features, it may make the necessary code modifications and additions itself. It may also choose to contract a business that has experience with a particular FOSS application in order to make the

[63] See Ubuntu CDs, *supra* note.

[64] Karim Lakhani & Jill Panetta, "The Principles of Distributed Innovation" in Hind Benbya and Nassim Belbaly, eds., *Successful OSS Project Design and Implementation* (Surrey, England: Gower Publishing Limited, 2011) [*Distributed Innovation*] at 14-15.

changes on its behalf. Core developers of the software are often willing to make the required adjustments; FOSS project teams regularly undertake work to advance a project through cost-recovery agreements with multilateral organizations.

Once new features are added, there are numerous reasons why a business or government organization may choose to redistribute the additions. For example, pushing the feature additions back into the main development stream means that the company or government organization will not need to re-integrate the changes itself into any future versions of the software. For a discussion of other benefits, please refer to the section below on benefits.

5.2.2 Use of FOSS in Other FOSS Projects

When creating a new FOSS software application, it is rare for software developers to write all components of the software from scratch. Most often, developers take advantage of existing software "libraries" and components that provide a base layer of functionality. These libraries and components may perform specialized mathematical calculations, handle database transactions, render user-friendly graphical elements, or conduct a myriad of other tasks. For example, the Apache Commons project provides more than forty re-usable Java software libraries that take care of a variety of tasks that developers commonly encounter.[65] These Apache Commons libraries are all available as FOSS, under the OSI-approved and FSF-approved Apache 2.0 licence.[66]

In most cases, it would be overly burdensome to require each user of a software application to independently download each and every library on which a software application depends. For this reason, developers usually bundle libraries along with the software application. This involves a distribution of the library whenever the developer distributes the application.

Occasionally, where an existing library needs extensive modification for a particular task, a developer might also copy and paste the code, make changes, then directly integrate the modified code into the project. In this case, the project constitutes a derivative work and its distribution often invokes hereditary licensing obligations.

[65] Apache Software Foundation, "Apache Commons" (2012), <http://commons.apache.org>.

[66] Open Source Initiative, "Open Source Licenses by Category" (accessed September 2012), <http://www.opensource.org>; Free Software Foundation, "Various Licenses and Comments about Them" (accessed September 2012), <http://www.gnu.org>.

Example – Distribution under Hereditary Licensing: What obligations apply?

Project Manager: "Dave, our client needs to check 50 000 points to determine whether they lie within city boundaries. They need an automated tool."

Task Resolution: Dave modifies the *inpolygon* function in GNU Octave's *inpolygon* function so that it can read from a spreadsheet of geodetic points (not a best coding practice!).

Original: *inpolygon(X,Y,xv,yv)*
Modified: *inpolygon(CSV_filename,xv,yv)*

What legal terms apply? Dave looks over the GPLv3 license that applies to GNU Octave and finds that the following apply:

- ✓ **Disclaimers of warranty and liability?** Yes, as set out in s. 15 & 16 of the GPLv3.
- ✓ **Notice obligations?** Yes, as set out in s. 4 & 5(b) of the GPLv3 ("Conveying Verbatim Copies").
- ✓ **Hereditary obligations?** Yes, as set out in s. 5(c) of the GPLv3 this obligation engages considering:
  (1) the particular licence: Here, under the GPLv3, hereditary obligations apply to all derivative works whenever you "convey" the program.
  (2) the type of distribution: Sending the program to the client or making it available for the client to download would constitute a "conveyance" under the GPLv3.
  (3) the extent of integration: Given that Dave copied and pasted the original function, the modified program is a derivative work (it is a "work on which the Program is based").
- ✓ **Source code obligations?** Yes, as set out in s. 6 of the GPLv3 ("Conveying Non-Source Forms").

Legal Compliance: Dave must comply with notice obligations and source code obligations in the same manner as when he distributes the original software (*supra*, p. 27).

To comply with the hereditary license obligations, Dave must license his modified version under the GPLv3 licence.

5.2.3 Use of FOSS in Restricted-Source Software Projects

Likewise, restricted-source software vendors rarely start their product development process from scratch. They commonly license commercial libraries and components from other companies. They also commonly use FOSS libraries and components, such as the aforementioned Apache Commons utilities.

Restricted-source vendors are able to use FOSS libraries without releasing their own code as long as the particular FOSS licence permits this. As previously discussed, permissive licences such as Apache 2.0 do not impose any obligation on the redistributors to release their own code. Even some hereditary licences – foremost, the Lessor GNU Public Licence (LGPL) – do not impose a hereditary obligation for merely linking to another library.

Example – Distribution under Hereditary Licensing: What obligations apply?

Project Manager: "Dave, we now need to provide our city boundary check as a tool in our GeoXToolbox product. We sell this under a restricted-source commercial licence and do want the GPLv3 to apply."

Task Resolution: Dave finds an OSGeo library called GEOS (Geometry Engine, Open Source) which contains the function he needs: *SimplePointInRing::isInside().* To implement the required functionality, he calls this function from the GeoXToolbox product.

What legal terms apply? Dave finds that GEOS is licensed under the GNU Lesser General Public License v2.1 (LGPLv2.1), with the following terms applying:

- ✓ **Disclaimers of warranty and liability?** Yes, similarly to the GPLv3 (see example *supra,* p. 29).
- ✓ **Notice obligations?** Yes, similarly to the GPLv3.
- ✗ **Hereditary obligations?** No, as set out in s.6 of the LGPLv.2.1, the hereditary obligation does not engage considering:
  - (1) the particular licence: Here, under the LGPLv2.1, hereditary obligations apply to all derivative works, except when merely linking to libraries; the obligations trigger whenever you "convey" the program..
  - (2) the type of distribution: Sending the program to the client or making it available for the client to download would constitute a "conveyance" under the LGPLv2.1.
  - (3) the extent of integration: As long as Dave only links to the GEOS function library – and does not directly modify the GEOS source code or copy and paste it – Dave's implementation falls within the linking exception of the LGPLv2.1 (it is a "work that uses the library").
- ✓ **Source code obligations?** Yes, similarly to the GLPv3 (with respect to the GEOS library as included in the GeoXToolbox, but the obligation does not apply to the rest of the program).

5.3 Typical Scenarios of FOSS Distribution of New Software (Use Case D)

5.3.1 Non-competing Products

When developing entirely new software – whether it is an application or a software library – a business or government organization needs to make a choice between keeping the source code restricted and releasing it as FOSS. Governments and non-profit organizations may wish to release software as FOSS to provide a range of additional benefits to FOSS communities, businesses, and the general public. Software vendors and other businesses often release software as FOSS where it is ancillary to their core business. They thereby benefit from others collaborating on the software, spreading out the development costs, without reducing their market share for their core competencies.

5.3.2 Alternative Business Models

With increasing popularity, software developing companies are also creating alternative business models with FOSS as a core part of the product strategy.[67] These business models include:[68]

Model	Description
Loss leader model	A business distributes a software package for free, as free and open source software, to help improve the company's position and name recognition in the software market. This helps the company sell the restricted-source offerings (or other goods or services) in its product line.
Sell it, free it	Under this model, a business distributes code for previous releases of its product as FOSS. The paying customers essentially pay a premium to use the latest version, at the earliest possible time.
Dual licensing	In this model, a business simultaneously distributes a software application under both a commercial restricted-source and a FOSS licence. The FOSS licence is generally one that tends to limit use by commercial vendors (such as a GPL licence). Thus, other commercial companies that want to integrate the software into their own applications must pay for the commercial licence.
Widget frosting	With this business model, a business openly releases software that compliments a hardware device. For example, a printer manufacturer might open-source its printer drivers to encourage Linux users to integrate the driver and use its printers.

[67] See Benno Luthiger & Carola Jungwirth, "The Chase for OSS Quality: The Meaning of Member Roles, Motivations, and Business Models" in Sulayman Sowe, Ioannis Stamelos & Ioannis Samoladas, eds., Emerging Free and Open Source Software Practices (Hershey: IGI Publishing, 2008) at 155-159.

[68] *Ibid*.

Model	Description
Service enabler	A business monetizes the service provided using open source software – most often, a web service such as file conversion or cloud-based file hosting. Where a FOSS community helps improve the software, this makes the service offering more attractive.
Support service model	Under this model, a business distributes software under a FOSS licence to maximize distribution and build a large base of users. The business profits from support services that it provides to the users (installation, resolving technical issues, maintenance, feature requests, etc.).

5.4 Policies and Perspectives on FOSS Distribution

Developers who contribute to FOSS and distribute it come from wide and varied environments across the public sector, private sector and academia. Studies show numerous and diverse motivations for FOSS participation, ranging from economic gain to a sense of civic duty towards social and technological advancement.[69] Risks and drawbacks of FOSS participation also widely differ, depending on the context.

5.4.1 Benefits

Collaboration to Lower Development Costs

Distributing code as FOSS allows others to join in on the development effort, making a project a joint effort amongst multiple companies, public sector workers, and individual volunteers.[70] Companies regularly invest a share of resources to participate in FOSS projects: 40% of developers in OSS communities are paid to participate.[71]

When collaborating to develop new features or creating a new project built on existing FOSS, a company or organization also leverages all the work that collaborators have previously invested. If distributing modifications as FOSS, prior work can even include code and libraries under hereditary licences (which a company or organization can only ever distribute if its project is FOSS as well).

[69] See generally Andrew Schofield & Grahame S. Cooper, “Perceptions of F/OSS Community: Participants’ Views on Participation” in Sulayman Sowe, Ioannis Stamelos & Ioannis Samoladas, eds., Emerging Free and Open Source Software Practices (Hershey: IGI Publishing, 2008) at 72-76.
[70] *Handbook on OSS*, *supra* note at 190.
[71] *Distributed Innovation, supra* note 64 at 15.

Long-term Feasibility

Releasing software as FOSS to encourage others to collaborate on it can also help ensure the viability of a project into the future. For example, an organization or company may develop a software tool with a fixed and limited research and development budget. For the project to continue and grow, the participation of other collaborators is necessary. In the case of public sector organizations releasing FOSS, project "buy-in" by a company has proved particularly beneficial in securing long-term viability.[72]

Reputation Building

For individuals participating in FOSS, a common motivating factor and concrete benefit is building a good reputation. In some cases, the goal is to exhibit skills and talents to employers; in other cases, it is to earn status with a community.[73]

Most FOSS projects are open for any developer to contribute. Thus, participants are able to develop and demonstrate their talent to potential employers. Not only can participants add their involvement to a C.V., but employers also sometimes screen and hire talent directly from a pool of the developers contributing to a FOSS project.[74] These developers may have proven competencies in desirable areas of expertise.

A culture of "meritocracy" popular within many FOSS projects also serves to benefit contributors with status and privilege.[75] In general, the strongest contributors tend to have the most control over the project – for example, in deciding upon priorities for new features.[76]

Of course, reputation and good will are also extremely important to commercial contributors. Company participation in FOSS can help build a positive reputation amongst other developers and amongst FOSS users.

Public Benefits

FOSS can align well with the role of public sector agencies in providing wide benefits to the public-at-large, such as in maintaining society's technological infrastructure and helping it evolve.[77] The public sector has a long history of participation in open source software, with

[72] See Andrea Bonaccorsi et al., "Firms' Participation in Free/Open Source Projects: Theory and Preliminary Evidence" in Hind Benbya and Nassim Belbaly, eds., *Successful OSS Project Design and Implementation* (Surrey, England: Gower Publishing Limited, 2011) at 38.

[73] *Karim Lakhani & Jill Panetta*, "The Principles of Distributed Innovation" in Hind Benbya and Nassim Belbaly, eds., *Successful OSS Project Design and Implementation* (Surrey, England: Gower Publishing Limited, 2011) at 14-15.

[74] *Ibid.*

[75] See Ross Gardler & Gabriel Hangaru, "Meritocratic Governance Model", *OSS Watch* (14 February 2012), <http://www.oss-watch.ac.uk/>.

[76] *Ibid.*

[77] *Open Source Technology and Policy, supra* note at 309.

public investment in open source dating back to when the U.S. government funded the development of the initial internet infrastructure, including numerous underlying FOSS components.[78]

A newer phenomenon is the emergence of social enterprises and entrepreneurial interest in making governments work better, particularly at the local level.[79] Civically-engaged technologists develop FOSS applications to solve social issues such as bureaucratic delays. Examples include SeeClickFix, a location-based web platform that allows residents to document neighborhood concerns and suggest improvements (with respect to garbage collection, graffiti, potholes, etc.).[80] In fact, in Canada a key motivation in the launch of a federal government open data portal was to provide "Canadian entrepreneurs with business opportunities that will benefit them and all Canadians".[81] Although not all software projects with broader social aims are FOSS, the benefits of FOSS in improving the overall availability of re-usable technology in society often aligns well with the aims of these social entrepreneurs.

Even outside of social enterprises, the release of FOSS by public sector organizations can help stimulate innovation in the private sector. It enables companies to create specialized offerings built on FOSS, even where such software might otherwise be too expensive for the company to develop in-house.

FOSS can also help maximize overall economic efficiency within society. Where software is freely available, and where anyone can add any new features needed, companies can make use of these existing resources rather than expend efforts duplicating an existing project.

Security

As discussed with respect to FOSS use, FOSS distribution likewise has security pros and cons. FOSS puts more eyes on the code to fix security issues but, at the same time, it makes the code available to those with malicious aims.

5.4.2 Drawbacks & Risks

Lack of Direct Licensing Revenue

The fact that anyone can redistribute FOSS for free is an obvious barrier for many software vendors who are not engaging the FOSS business model. Other than nominal charges for distribution on physical media, it is generally not feasible to profit from FOSS using the traditional business model of directly selling software licences.

[78] *Ibid.*
[79] Brian Braiker, "The open source problem solvers creating government 2.0", *The Guardian* (3 May 2012), <http://www.guardian.co.uk>; see also Open North, "About" (accessed September 2012), <http://opennorth.ca>.
[80] See ClickFix, "About SeeClickFix" (accessed September 2012), <http://seeclickfix.com>.
[81] Treasury Board of Canada Secretariat, "Minister Clement Announces Expansion of Open Data Portal" (28 June 2012), <http://news.gc.ca>.

The lack of direct licensing revenue can also pose a barrier in public sector organizations where "cost recovery" policies are in place. Even though FOSS may aid numerous public service goals, these do not necessarily help to offset the costs and risks of developing the software.

Community May Not Coalesce

There are many examples of thriving FOSS projects such as Linux, the Apache web server and Firefox. These projects involve active communities with hundreds, and in some cases thousands, of software developers.[82] However, there is also a relatively high count of FOSS projects where active development has ceased.[83] On the popular online FOSS hub of SourceForge.net, there are more than 100 000 software projects but a majority of these do not have an active community and are not making any substantial progress towards a finished release.[84] Of course, many of these projects are likely those of individual developers who registered a project but made no subsequent efforts or headway in developing a community. In any case, the numbers do suggest a cautionary tale: to properly execute a FOSS project, you should be prepared to invest the resources to see a project through a first release and any lag in community involvement.

Legal Complexities

Releasing software as FOSS usually requires a careful legal consideration of the licences, especially when the software includes libraries and code from multiple sources. Although restricted-source licences also require careful legal scrutiny, many FOSS licences are long and contain legal complexities with which in-house legal staff may not be familiar. For example, the GPLv3 spans fifteen pages of legal text and contains numerous specialized legal clauses which legal experts and academics frequently study and debate.[85]

Patent Liability Risk

Opinions are equally divided on whether releasing software as FOSS increases or decreases the risk of patent liability. On the one hand, distributing your code as FOSS opens it up to further scrutiny. Others can look through the code and attempt to find patent infringements. On the other hand, collaborators on FOSS projects may help re-engineer around patents as soon as an infringement is discovered. The potential adverse consequences on public relations and good will in launching a patent lawsuit against a FOSS community can also have a strong deterrent

[82] For example, 1 980 contributors have participated in developing Mozilla Firefox (Ohloh, "Project Summary: Mozilla Firefox" (accessed September 2012), <http://www.ohloh.net>).

[83] Austen Rainer and Stephen Gale, "Little Fish in a Big Pong: A Comparison of Active SourceForge OSS Projects with Very Popular Non-SourceForge OSS Projects" in Hind Benbya and Nassim Belbaly, eds., *Successful OSS Project Design and Implementation* (Surrey, England: Gower Publishing Limited, 2011).

[84] *Ibid.* at 172-175.

[85] GPLv3, *supra* note (ODT version).

effect.[86] Non-profit organizations such as the Software Freedom Law Center also offer resources to help FOSS projects defend against patent infringement suits and invalid patent claims.[87]

Another drawback is that individual volunteers rarely have defensive patent portfolios (these are patent portfolios that major software companies build up in order to leverage them in countersuits, or threats of countersuits, against anyone who might attempt a patent lawsuits against them, often resulting in a patent litigation stalemate). However, by the same token, going after individuals is also expensive for patent litigators. Most individuals alone do not have enough net worth to make a patent lawsuit against them worthwhile.[88]

5.5 Best Practices for FOSS Distribution

5.5.1 Deciding to Distribute Software as FOSS

A decision on whether to license software as FOSS should always start with an assessment of the business requirements and the aims of the project. The business requirements will greatly impact the weight that you should give to various benefits and drawbacks.

Companies and organizations release software as FOSS at many different stages of the development cycle. In some cases, software has seen many releases and iterations before it becomes FOSS. In other cases, software may commence its life as a collaborative FOSS project amongst several parties. However, it is generally a good practice to have a plan for the initial project architecture before starting distribution as FOSS. The plan may be as simple as having several collaborators start working on the architecture when the project commences. Where no plan exists, however, developers working on different pieces of the project might run into difficulties integrating their respective pieces or working into a cohesive application.

5.5.2 Choosing a Licence

You will not always have a choice as to which licence you apply. Where a hereditary licence obligation is in force, you need to license your code under the same licence – see the section on *Due Diligence / Licence Management*, below. As well, even if you are not under a strict legal obligation to apply a particular licence, you may still wish to adopt the same licence as an existing software project or community in order to become involved with it.

[86] Heather Meeker, *The Open Source Alternative* (Hoboken: John Wiley & Sons Inc., 2008) at 96.
[87] See Software Freedom Law Center, “SFLC's Services” (accessed September 2012), <http://www.softwarefreedom.org>; see also Electronic Frontier Foundation, “Patent Busting Project” (accessed September 2012), <http://www.eff.org>.
[88] Meeker, *supra* note at 95.

Where you distribute a project consisting entirely of your own code, or consisting of your own code along with permissively-licenced code and code which does not engage hereditary obligations, you can choose the FOSS licence yourself. You may even opt to dual-license your code, perhaps if you wish to participate in two different communities that use different licensing norms.

The licence you choose should reflect your business requirements. All common FOSS licences can be adopted for works by government, industry, or the education sector – you need to look at particular project aims.

Overall, licensing decisions tend to involve one primary and consequential decision: whether to apply a hereditary or permissive licence:

- **Permissive licences** maximize the scope of downstream users (with broad appeal to the entire private sector); while
- **Hereditary licences** are appropriate in cases where it is important to receive back downstream changes, or where it is important to ensure that work built on an initial investment remains open and free. Hereditary licences can also put a focus on benefiting other private-sector businesses that provide services and support.

The following chart details other key differences in this decision:

	Permissive	Hereditary
Beneficiaries of the FOSS release	Everyone: commercial software vendors, support services, etc.	Everyone, but only where they are willing to release their software as FOSS, under the same licensing terms as were granted to them (note that some restricted-source software vendors absolutely prohibit hereditary licences such as the GPL).
Beneficiaries of downstream code changes	The whole community, but only where the business (or other developer) chooses to contribute modifications back under the permissive licence.	The whole community in every case where a business, organization, or individual distributes the modifications, as the licence then mandates releasing the changes under the same FOSS licence.
Licence complexity	Often very simple and understandable (e.g., popular "2-clause BSD").	Relatively complex, requiring careful legal analysis (and some risk of misinterpretation).
Interoperability	Permissively-licenced code can be included in projects under hereditary licences, other permissive licences, or restricted-source licences.	Hereditary-licenced code cannot generally be included in a project under any other single licence.

5.5.3 Due Diligence / Licence Management

Managing Licence Obligations

Where hereditary licensing obligations apply, you do not have the benefit of being able to choose a licence. Your code – to the extent specified in the hereditary licence – must come under the exact same licence when you distribute it (or, if the licence permits, a later version of the same licence). In such a case where you encounter a hereditary licensing obligation, there are three ways to comply:

1. Do not distribute your software;
2. License your software under the exact same licence (or, where the licence permits, a compatible licence); or
3. Re-implement parts of your software such that it does not include any code or libraries that come under a hereditary licence (or, at least, ensure that your code does not integrate tightly with hereditary code such that the obligation engages).

Restricted-source software vendors typically opt for solution three when they notice inadvertent hereditary code, given that they need to distribute their software to paying customers but do not usually want to open up the rest of their source code to others. It is therefore important for these vendors to conduct a “due diligence” audit on their software projects, checking that their software does not include any FOSS code or libraries that engage a hereditary licensing obligation.

Likewise, organizations and businesses releasing their code as FOSS should also conduct due diligence audits. Although they are already applying a FOSS licence, a hereditary licensing obligation generally imposes a requirement to licence the code under the exact same licence. Thus, releasing code under a different FOSS licence may not always comply. It may be necessary to dual-license your own code under the other hereditary licence, or, in a similar manner to the restricted-source context, ensure that the software does not include any code or libraries that engage the hereditary obligation.

Although hereditary obligations pose the strictest set of parameters, organizations and businesses must also ensure they comply with other licensing terms. For example, they must comply with notice requirements and obligations to distribute the original source code. A due diligence audit ensures that this compliance is in place.

There are two general methods to conduct a due diligence audit: (1) provenance checking and (2) code scanning.[89]

[89] Meeker, *supra* note at 72.

Provenance Checking

Provenance checking involves maintaining a careful audit trail: the developers maintain internal records of what code is in the project, how that code is used, and what licence applies to each element.[90] Some build automation tools such as Maven (which developers use to automate code compilation and deployment) help with functionality to indicate, track, and report licences in a project, thereby assisting and standardizing the task of keeping records.[91]

Looking at the internal audit records, a licensing expert can check a project for compliance either when a developer adds a new external element or upon release of the software (or both). A fresh legal analysis of the licence text is not required for each and every new library imported into a project. Once a licence manager approves use of a particular licence within a project, developers can generally safely use other libraries under the same licence, as long as they use them in a similar manner. For example, a business or organization might establish a policy for a project that: grants automatic approval for specific permissive licences including BSD, MIT and Apache; grants approval for weak hereditary licences such as the LGPL on a case-by-case basis; and grants approval for strong hereditary licences, such as the GPL, only after a careful and thorough legal analysis.

Automated Code Scanning

In many cases, provenance checking should prove sufficient for smaller projects. However, it may prove impractical for larger companies or larger projects. A large company often owns code purchased from other parties, or code received from acquisitions and mergers, that may have no accurate licence audit for the code. In this case, it is best to use automated code scanning tools that search through the entire code base to determine the licences that apply.

Automated code scanning utilities search text files and embedded code comments that may indicate the licence applicable to a particular element of the software.[92] Some tools even compare the code itself against known third-party FOSS code.[93]

Two of the more sophisticated and popular code scanning tools that implement both of these methods are the software packages available from Ottawa-based Protecode and U.S.-based Black Duck Software. These companies also provide professional code auditing services.[94]

[90] *Ibid.*
[91] See Apache Software Foundation, "POM Reference" (2012), <http://maven.apache.org>; see also SoftwareEntwicklung Beratung Schulung, "Maven License Verifier Plugin" (2011), <http://khmarbaise.github.com/Maven-License-Verifier-Plugin>.
[92] See e.g. Protecode, "Enterprise Analyzer" (2012), <http://www.protecode.com>.
[93] *Ibid.*
[94] Protecode, "Protecode Certified Audit Service" (accessed September 2012), <http://www.protecode.com>;Black Duck Software, "Audit Services Overview" (accessed 2012), <http://www.blackducksoftware.com>.

Of course, while these tools can prove highly useful, it must be kept in mind that the results do not provide certainty that the source code is under only the reported licences, nor that it is free of copyright or patent infringements. An inherent limitation of the audit is that it can only compare the client's source code to a wide, but not exhaustive, collection of other source code repositories. It may not detect copyrighted source code from restricted-source vendors, or source code from smaller FOSS projects.

Where possible, businesses and organizations may also wish to reduce their risks and ease the auditing task by using software libraries from trustworthy organizations that have already audited the library code. For example, the Eclipse Foundation tries to carefully check and manage licences for all contributions to the projects that it manages.[95] Similarly, the Open Source Geospatial Foundation (OSGeo) requests projects to undergo a "Code Provenance Review" before they can become member projects of the foundation.[96]

5.5.4 Patent Management and Other Legal Issues

All of the FOSS legal considerations which apply when using FOSS equally apply when you distribute FOSS. For code that you distribute, the lack of a disclaimer and warranty can work in your favour. The lack of a choice of forum or choice of law clause generates equal legal uncertainty for all parties. In addition to these legal issues, FOSS distribution raises concerns related to patents.

By licensing your code under a FOSS licence, you may either implicitly or explicitly license the patents you own if any of the code implicates them. It is important to understand the nature and scope of the patent licences you grant.

Patent Licence Scope

The treatment of patents varies greatly throughout different FOSS licences. The traditional approach – still seen in popular permissive licences such as BSD and MIT – is an implicit patent licence: the licence makes no specific mention of patents, but rather the stated right to use the software implicitly grants the licensee permission to "use" any relevant patents held by the licensor.

An implicit patent licence almost certainly grants others the right to use the original code as distributed, including where such use implicates patent held by the licensor. However, the scope of an implicit patent licence becomes less clear when downstream parties modify the original code. If the modifications change the typical "use" of the software, does the original patent licence still cover a use that is different from what the licensor originally intended? If a new use

[95] Eclipse Foundation, "Commiter Due Diligence Guidelines" (2008), <http://www.eclipse.org>.
[96] Open Source Geospatial Foundation (OSGeo), "Project Status Template Version 1.1" (accessed September 2012), <http://osgeo.org>.

infringes a different patent held by the original licensor, does the broad grant of rights to make modifications also end up licensing this other patent?

Most likely, the answer to these two questions in “no”: the implicit patent grant often covers only uses implicated by the licensor's original contributions, but not other uses that additional features and modifications might involve.[97] Most modern FOSS licences attempt to increase clarity and legal certainty by making this scope limitation explicit. For example, the Apache Version 2.0 licence provides:

> *Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted.*[98]

Under this clause, the patent licence only extends to uses of the software applicable to *existing* contributions. Where further downstream contributions alter the software's use, the original patent licence may no longer apply.

Although a FOSS licence could feasibly grant a broader patent licence that would cover downstream modifications, no popular licences presently take this approach. Not even the highly freedom-assertive GPLv3 licence makes such a grant. Given the nearly limitless number of ways that additional features could alter the typical use of a software application, such a broad patent licence is likely untenable for most businesses managing a patent portfolio.

Therefore, as a best practice, whenever you modify FOSS you should consider whether the modifications change the use of the software in a way that might implicate other patent licences, or in such a way that existing patent licences may not cover the new use.

[97] No Canadian jurisprudence directly addresses this issue but see EC, *Commission Decision of 21.01.2010 declaring a concentration to be compatible with the common market and the functioning of the EEA Agreement* (Case No COMP/M.5529 - Oracle/ Sun Microsystems), C(20120) 142 final at para 733.

[98] Apache Software Foundation, “Apache License, Version 2.0” (2004), <http://www.apache.org>.

Patent Termination and Retaliation Clauses

Many FOSS licences attempt to protect the software against patent infringement lawsuits by including automatic-termination clauses. These clauses trigger whenever a licensee alleges that any part of the software infringes his or her patent. For example, the Apache Version 2.0 licence succinctly states:

> *If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.*[99]

These triggers differ amongst licences. For example, unlike the Apache Version 2.0 licence, the Mozilla Public License Version 2 (MPLv2) explicitly allows parties to defend themselves with patent infringement counterclaims and crossclaims, all without triggering the termination clause.[100]

Some licences also include broader retaliation clauses – that is, a broader termination of rights. The Apache Version 2.0 patent termination clause, set out above, only terminates patent licences. The MPLv2, on the other hand, terminates all rights under both copyright and patent law.[101]

When involved in patent infringement litigation – whether initiating an originating action or a counterclaim – it is important to carefully assess the impact this could have on any FOSS that you use or contribute towards.

5.5.5 Managing Project Participation

Open source software often brings together a disparate community of developers, ranging from volunteer hobbyists to commercial enterprises. In the absence of a formal management and communication structure as found in a unified corporate development environment, FOSS communities use a variety of techniques to self-manage their projects in this environment.[102]

[99] *Ibid.*
[100] MPLv2, *supra* note 7, s. 5.2.
[101] *Ibid.*
[102] Ruben van Wendel de Joode, Hans de Bruijn & Michel van Eeten, “Software Development and Coordination Tools in Open Source Communities” in Sulayman Sowe, Ioannis Stamelos & Ioannis Samoladas, eds., *Emerging Free and Open Source Software Practices* (Hershey: IGI Publishing, 2008) at 96-98.

Popular project management techniques for FOSS include:[103]

Technique	Description
Project managers	Project managers help organize the larger community of developers. In many cases, the *de facto* project manager is the person who makes the first lines of code. Some communities, such as Apache, vest their leadership in a board of directors. The role of a project manager in an open source project is much more limited than in a proprietary setting: unless the manager is paying someone, she or he cannot always delegate tasks within a specific timeline. Moreover, if a project manager imposes too much control, it can be resisted by the group. Thus, the role is more one of controlling debates, and facilitating task co-ordination.
Software modularity	A community can use a "divide and conquer" approach to divide the software project into many smaller and less complex parts. The aim is for these modules to be easy to understand and contain as few interdependencies as possible. This allows an individual developer or small group of developers to separately work on a part of the project in an individualistic fashion, based on their own individual choices and preferences.
Issue tracking systems (e.g., Bugzilla, Redmine, Chiliproject)	These systems keep track of software bugs, feature requests and other tasks, from the initial report to the successful testing of a resolution.
Versioning systems (e.g., Git)	These systems permit remote access to a central repository of source code and allow multiple developers to work on the project at the same time. These systems maintain an archive of all older versions of the software code, thereby protecting the system against potential damage from mistakes that any contributor could inadvertently introduce. These systems also provide tools for merging together code changes where developers work on a file or multiple files simultaneously.
Coding style guidelines	Coding guidelines prescribe the way in which developers should style and lay out source code, making it easy for collaborators to read and edit each other's work.

5.6 Interoperability and Licence Alignment

FOSS distribution, as opposed to mere use, requires more careful attention to licence interoperability issues. First, you may encounter a *soft* incompatibility, such that you can still technically redistribute a combined software work, but not necessarily under the licence of your choice (you face a licence alignment barrier). Second, you may encounter a *hard* incompatibility which could absolutely prohibit you from combining and distributing a combined software work.

5.6.1 Soft Licence Incompatibilities

A *soft* incompatibility can arise when a developer integrates code or a library under a hereditary licence with other permissively-licensed software. For example, consider a hypothetical company GeoX that uses the cartographic projection library PROJ.4 in its products. GeoX runs

[103] *Ibid.*

into the need for a specialized map projection. One of its developers searches Google Code, finds a math implementation under a GPLv3 licence, and integrates it into a local version of PROJ.4

Up until this point, no licence violation has occurred. However, GeoX can only distribute its new version of PROJ.4 under a GPLv3 licence. In some business cases, this may be acceptable. For example, if there is only a very loose coupling between the PROJ.4 library and other components of the system, GeoX can distribute its new PROJ.4 under GPLv3 and not the rest of its product. GeoX can continue to distribute its own software under the licence of its choice, including under a restricted-source licence, as long as it distributes the source code for the new PROJ.4 component.

However, GeoX could not integrate its changes back into the primary PROJ.4 development stream, which uses a permissive MIT licence. Even though GeoX can distribute the changes under GPLv3, the whole PROJ.4 project would have to also move to the GPLv3 licence, which the numerous other developers on the project are unlikely to do. This means that GeoX would have to reintegrate its in-house changes if it moved to a newer version of PROJ.4.

More importantly, if the coupling between PROJ.4 and other components of GeoX's product were integrated so tightly such that the other components constituted derivative works, GeoX could no longer sell its product under only a restricted-source licence. It would have to also release the project under GPLv3 – which is clearly not likely an option for GeoX if this is a major source of revenue. In that case, it may be better for GeoX to write its own solution.

5.6.2 Hard Licence Incompatibilities

A *hard* incompatibility can arise when you integrate code or libraries under two different hereditary licences. Given that hereditary licences only allow you to license derivative works under the exact same licence, two conflicting “same licence” requirements can place an absolute barrier on any redistribution.

In this context, other than abandoning use of one of the conflicting components, your only option for distribution is to ask the developers of one of the components to dual-license their code under the other hereditary licence, or under a permissive licence. This is a difficult and often insurmountable hurdle when hundreds of different developers may have contributed to a project. Unless all participating developers have assigned their copyright to a single organization managing the software project, they all need to independently agree to their contributions coming under the different licence.

This difficulty only arises with hereditary licences. Under all popular permissive licences, you can freely combine any permissively-licensed works with one another without any compatibility difficulties. You can also always combine permissively-licensed works into software under one specific hereditary licence.[104]

5.6.3 Restrictive (Non-FOSS) Licence Incompatibilities

Restrictive (non-FOSS) licences can also create incompatibilities. Companies and developers should take particular care with software licences that are "shared source", but not free or open source. For example, the "Microsoft Shared Source CLI, C#, and JScript License" permits modification and distribution, but not for commercial purposes.[105] This is not a FOSS licence because the non-commercial requirement complies with neither the Free Software Foundation's definition of Free Software, nor the Open Source Initiative's definition of Open Source Software.

These licences imposing restrictions such as non-commercial use remain incompatible with all FOSS licences, given that all FOSS software licences *must* permit all uses, including commercial use. Article 6 of the OSI Open Source Definition states:

> *6. No Discrimination against Fields of Endeavor*
>
> *The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.*[106]

Thus, were you to combine software under the Microsoft Shared Source CLI with any FOSS licence, the non-commercial clause would conflict with the FOSS licence and you could not redistribute the software.

[104] The only exception amongst common licenses is, arguably, an incompatibility between the older Apache licenses (1.0/1.1) and the GPL (see Free Software Foundation, "GPL-Compatible Free Software License" (accessed September 2012), <http://www.gnu.org>).
[105] Microsoft, "Microsoft Shared Source CLI, C#, and JScript License" (accessed September 2012), <http://www.microsoft.com>.
[106] *OS Definition, supra* note 3.

6. Conclusions

The overall process for obtaining or distributing FOSS is similar, and equally complex, as the restricted-source context. When obtaining FOSS, you still must assess business needs, overall costs, and long-term support, maintenance and training. When distributing software as FOSS, you need to assess how this fits into your business model and you need to conduct a careful legal assessment – just as a vendor always carefully analyzes the terms of a restricted-source licence for a new purchaser.

Of course, although the overall assessment processes remains similar to the restricted-source context, differences abound in the specific considerations that arise in the FOSS context. When it comes to a legal assessment for using FOSS, the inability to "flow through" risk to the licensor generates a need to assess this level of risk and consider alternatives such as third-party insurers. When distributing FOSS, it becomes imperative to conduct a due diligence assessment of licence obligations and licence interoperability within your software. If you are distributing your own software, the choice of licence raises not only issues about how you can maintain your revenue streams and minimize your legal risks, but also about the types of communities you want to reach and with which you want to interact.

With organizations such as OSGeo continuing to host more projects, there are a quickly growing number of FOSS software tools available to those working with geospatial information. None of these are likely to be one-size-fits-all solutions and most organizations will probably find themselves using a mix of FOSS tools and restricted-source tools. However, organizations do not at least assess FOSS tools and consider integrating them into their processes are likely missing out on opportunities for collaboration and cost savings.

Appendix 1: Glossary

Acronym	Term	Definition
	Copyleft licence	See hereditary licence.
	Copyright	The set of rights granted under Canada's Copyright Act, RSC 1985, c C-42 (or, internationally, under the Berne Convention) to authors of literary, musical and other artistic works. Copyright covers software source code under the category of literary works.
FOSS	Free and Open Source Software	A term used to inclusively refer to both open source software and free software.
	Free Software	Software released under a licence that grants everyone broad rights to freely run the software, modify it, and redistribute it. Under the definition set out by the Free Software Foundation, a licence must grant four defined freedoms in order to meet the definition of free software; however, the term is often used more broadly and is commonly used interchangeably with the term open source software. Although free software is most often downloadable at zero cost, in this context “free” refers to “free as in freedom” (libre, not gratuit).
	Hereditary licence	A hereditary licence, also a “copyleft”, “share-alike”, or “reciprocal” licence, refers to a software licence which imposes an obligation to release any modified versions of the software under the exact same licence (whenever one distributes the modified version).
	Object code	Object code is the format of a computer program that a computer reads and runs. This is the only format that a computer user needs to execute a computer program, and it is therefore often the only format that vendors and distributors provide. However, object code is missing comments and other features of the original source code: it is therefore usually impractical for users and developers to modify software in this format.
OSS	Open Source Software	Software released under a licence that grants everyone broad rights to freely run the software, modify it, and redistribute it. Under the definition set out by the Open Source Initiative (OSI), a licence must meet ten criteria relating to these rights in order to meet the definition of open source; however, the term is often used more broadly and is commonly used interchangeably with the term free software.

Acronym	Term	Definition
	Patent	Refers to the time-limited protections granted under the Patent Act, RSC 1985, c P-4 (or analogous legislation in other jurisdictions) for new inventions that meet the criteria set out in the legislation.
	Permissive licence	A licence which grants broad rights and imposes few obligations onto users. The term is most commonly used in contrast to hereditary licences: unlike hereditary licences, permissive licences do not require modifications to be released under the same.
	Reciprocal licence	See hereditary licence.
	Restricted-source software	Refers to software that is not licenced as free software or open source software: in general, the licences for restricted-source software do not allow users to freely modify or redistribute the licenced programs. Most software licences purchased from vendors or distributors fall under this category of software.
	Source code	Source code is the human-readable format of a computer program that software developers write and modify in order to develop software. Depending on the computer language which a developer uses, source code may need to be compiled into object code before a user can execute it.

Appendix 2: References

Apache Software Foundation. “Apache Commons” (2012), <http://commons.apache.org>.

Apache Software Foundation. “Apache License, Version 2.0” (2004), <http://www.apache.org>.

Apache Software Foundation. “POM Reference” (2012), <http://maven.apache.org>.

BC Public Sector. “IM/IT Enablers Strategy v1.5” (2011), <http://www.cio.gov.bc.ca>.

Black Duck Software. “Audit Services Overview” (accessed 2012), <http://www.blackducksoftware.com>.

Black Duck Software. “How to Go Open: Successfully Open Sourcing Internal Software” (Legal Webinar Series, 2011).

Black Duck Software. “Ohloh” (accessed September 20120), <http://www.ohloh.net/>.

Bonaccorsi, Andrea et al., “Firms' Participation in Free/Open Source Projects: Theory and Preliminary Evidence” in Hind Benbya and Nassim Belbaly, eds., *Successful OSS Project Design and Implementation* (Surrey, England: Gower Publishing Limited, 2011).

Braiker, Brian. “The open source problem solvers creating government 2.0”, *The Guardian* (3 May 2012), <http://www.guardian.co.uk>.

Canonical. “Ubuntu Advantage Support” (accessed September 2012), <http://www.canonical.com/>.

Canonical. “CDs and DVDs” (accessed 2012), <http://canonical.com>.

Chung, Emily. “An Open Door for Open Source?”, *CBC News* (12 February 2009), <http://www.cbc.ca>.

Deek, Fadi & McHugh, James. “Open Source Technology and Policy” (New York: Cambridge University Press, 2008).

EC. *Commission Decision of 21.01.2010 declaring a concentration to be compatible with the common market and the functioning of the EEA Agreement* (Case No COMP/M.5529 - Oracle/ Sun Microsystems), C(20120) 142.

Eclipse Foundation, “Committer Due Diligence Guidelines” (2008), <http://www.eclipse.org>.

Electronic Frontier Foundation. “Patent Busting Project” (accessed September 2012), <http://www.eff.org>.

Fitzgerald, Brian & Suzor, Nic. “Legal Issues for the Use of Free and Open Source Software in Government”, (2005) *Open Source Software* 412

FreeBSD Project. “FreeBSD Copyright” (1992), <http://www.freebsd.org/>.

Free Software Foundation. “Free Software Definition” (2012), <http://www.gnu.org/>.

Free Software Foundation. “GNU Affero General Public Lessor Version 3” (2007), <http://www.gnu.org>.

Free Software Foundation. “GNU General Public License Version 3” (2007), <http://www.gnu.org>.

Free Software Foundation. “GNU Lesser General Public Lessor Version 3” (2007), <http://www.gnu.org>.

Free Software Foundation. “GPL-Compatible Free Software License” (accessed September 2012), <http://www.gnu.org>.

Free Software Foundation. “Various Licenses and Comments about Them” (2012), <http://www.gnu.org/>.

Gardler, Ross & Hangaru, Gabriel. “Meritocratic Governance Model”, *OSS Watch* (14 February 2012), <http://www.oss-watch.ac.uk/>.

Jelastic. “Software Stack Market Share: July 2012” (23 July 2012), <http://blog.jelastic.com/>.

Lakhani, Karim & Panetta, Jill. “The Principles of Distributed Innovation” in Hind Benbya and Nassim Belbaly, eds., *Successful OSS Project Design and Implementation* (Surrey, England: Gower Publishing Limited, 2011).

Luthiger, Benno & Jungwirth, Carola. “The Chase for OSS Quality: The Meaning of Member Roles, Motivations, and Business Models” in Sulayman Sowe, Ioannis Stamelos & Ioannis Samoladas, eds., *Emerging Free and Open Source Software Practices* (Hershey: IGI Publishing, 2008).

Meeker, Heather. *The Open Source Alternative* (Hoboken, NJ: John Wiley & Sons, 2008).

Microsoft, “Microsoft Shared Source CLI, C#, and JScript License” (accessed September 2012), <http://www.microsoft.com>.

Mozilla. “Mozilla Public License Version 2” (2011), <http://www.mozilla.org>.

National Research Council. “NRC Open Source Software Guidelines” (2012).

Netcraft. “August 2012 Web Survey” (2 August 2012), <http://news.netcraft.com/>.

Noyes, Katherine. “10 Reasons Open Source Is Good for Business”, *PCWorld* (5 November 20120), <http:www.pcworld.com>.

Ohloh, “Project Summary: Mozilla Firefox” (accessed September 2012), <http://www.ohloh.net>.

Open North. “About” (accessed September 2012), <http://opennorth.ca>.

Open Source Geospatial Foundation (OSGeo). “Project Status Template Version 1.1” (accessed September 2012), <http://osgeo.org>.

Open Source Initiative. “Open Source Licenses” (accessed 2012), <http://opensource.org/>.

Open Source Initiative, “Open Source Licenses by Category” (accessed September 2012), <http://www.opensource.org>.

Open Source Initiative. “The Open Source Definition” (accessed 2012), <http://opensource.org/>.

Open Source Software Directory, “About” (accessed August 2012), <http://www.opensourcesoftwaredirectory.com>.

Petry, Roger. “Free Software: Ideal for Saskatchewan” (2003) 3:5 CCPA Saskatchewan Notes 1, <http://www.policyalternatives.ca>.

Protecode, “Enterprise Analyzer” (2012), <http://www.protecode.com>.

Rainer, Austen & Gale, Stephen. “Little Fish in a Big Pong: A Comparison of Active SourceForge OSS Projects with Very Popular Non-SourceForge OSS Projects” in Hind Benbya and Nassim Belbaly, eds., *Successful OSS Project Design and Implementation* (Surrey, England: Gower Publishing Limited, 2011).

Red Hat, “Open Source Assurance FAQs” (accessed September 2012), <http://www.redhat.com>.

Regents of the University of Minnesota, “About” (2011), <http://mapserver.org/>.

Sale of Goods Act, RSO 1990, c S.1.

Schofield, Andrew & Cooper, Grahame. “Perceptions of F/OSS Community: Participants’ Views on Participation” in Sulayman Sowe, Ioannis Stamelos & Ioannis Samoladas, eds., *Emerging Free and Open Source Software Practices* (Hershey: IGI Publishing, 2008).

SoftwareEntwicklung Beratung Schulung. "Maven License Verifier Plugin" (2011), <http://khmarbaise.github.com/Maven-License-Verifier-Plugin>.

Software Freedom Law Center. "SFLC's Services" (accessed September 2012), <http://www.softwarefreedom.org>.

St. Amant, Kirk & Still, Brian. *Handbook of Research on Open Source Software* (Hershey: Information Science Reference, 2007).

Tilden Rent-A-Car Co. v. Clendenning, (1978), 83 DLR (3d) 400.

Treasury Board of Canada Secretariat, "Minister Clement Announces Expansion of Open Data Portal" (28 June 2012), <http://news.gc.ca>.

Treasury Board of Canada Secretariat, "Open Source Software Frequently Asked Questions" (2007), <http://www.collectionscanada.gc.ca/>.

U.S., *Uniform Commercial Code*.

Van Wendel de Joode, R., De Bruijn, H. & Van Eeten, M. "Software Development and Coordination Tools in Open Source Communities" in Sulayman Sowe, Ioannis Stamelos & Ioannis Samoladas, eds., *Emerging Free and Open Source Software Practices* (Hershey: IGI Publishing, 2008).

Ven, K., Nuffel, D.V., & Verelst, J. "The Migration of Public Administrations Towards Open Source Desktop Software: Recommendations from Research and Validation through a Case Study" in Sulayman Sowe, Ioannis Stamelos & Ioannis Samoladas, eds., *Emerging Free and Open Source Software Practices* (Hershey: IGI Publishing, 2008).

Wang, Julie & Robey, Dan. "Social Capital in OSS Communities: A Cross-Level Research Model" in Hind Benbya and Nassim Belbaly, eds., *Successful OSS Project Design and Implementation* (Surrey, England: Gower Publishing Limited, 2011).

Appendix 3: Legal Summaries of Common FOSS Licences

6.1 Apache License Version 2.0 [107]

URL: http://www.apache.org/licences/LICENSE-2.0.html
This licence is maintained by **Apache Software Foundation**.

6.1.1 Rights and Permissions

You are free to:

- **Use and reproduce** the work for your own purposes
- **Modify** the work and adapt it into your own projects
- **Distribute** the work and share it with others

Your freedoms under this licence apply to all **copyright** that vests in the work. However, you still must ensure that you do not infringe any:

- **Trade-marks**
- **Neighbouring rights (i.e. rights in a sound recording or performance)**
- **Database rights (where applicable under European law)**
- **Moral rights**

Patents held by the contributors are also **explicitly licensed**, but only for the types of use implicated at the time that each contributor made his or her contribution. If the software has since changed, or if you are modifying the software, beware of other patents that you might infringe.

6.1.2 Obligations

You must:

- **Retain the original copyright notices** and disclaimers to inform downstream users of the licence conditions that apply to the original work

[107] Licence summaries based on information generated from the CIPPIC Open Licensing Information Project (http://www.cippic.ca, not yet released). Please note that these summaries are intended as a general introduction to aspects of each license. They are in no way a replacement for a detailed analysis of each license text.

6.1.3 Disclaimers

The licensor makes **no guarantee** to you as to:

- **The quality of the work**. Keep in mind that the work may contain inaccuracies and errors. It may be entirely unsuitable for your purpose.
- **The clearance of rights**. It is possible that the licensor might not actually own all of the rights that the licence claims to grant. If this is the case, you could inadvertently infringe the copyright of a third party — and you will bear the full responsibility of this infringement.

The licence also contains a general **disclaimer of liability**. In most cases, you cannot sue the licensor, even if you suffer injuries or financial harm due to errors, inaccuracies, or faults on the part of the licensor.

6.1.4 Licence versioning

Even if the licensor releases a new version of the licence with different terms, you can **continue your use of the work under the original licence**.

However, if you retrieve an updated copy of the work, it may then come under the terms of the new licence.

6.1.5 Choice of Law and Forum

The licence **does not specify a forum** for any lawsuits. If a legal dispute arises between you and the licensor, courts will decide upon the location for the hearing based on how strongly the legal issue connects to the jurisdictions involved.

The licence **does not specify the particular law** that a court is to apply when resolving a dispute. To interpret the licence, courts will determine the appropriate law based on how strongly the legal issue relates to each jurisdiction involved.

6.2 BSD 2-clause “Simplified” License

URL: http://www.freebsd.org/copyright/freebsd-licence.html
This licence is maintained by **FreeBSD.**

6.2.1 Rights and Permissions

You are free to:

- **Use and reproduce** the work for your own purposes
- **Modify** the work and adapt it into your own projects
- **Distribute** the work and share it with others

Your freedoms under this licence apply to all **copyright** that vests in the work. However, you still must ensure that you do not infringe any:

- **Trade-marks**
- **Neighbouring rights (i.e. rights in a sound recording or performance)**
- **Database rights (where applicable under European law)**
- **Moral rights**

Patents held by the contributors are also **likely implicitly licensed**, but only for the types of use implicated at the time that each contributor made his or her contribution. If the software has since changed, or if you are modifying the software, beware of other patents that you might infringe.

6.2.2 Obligations

You must:

- **Retain the original copyright notices** and disclaimers to inform downstream users of the licence conditions that apply to the original work

6.2.3 Disclaimers

The licensor makes **no guarantee** to you as to:

- **The quality of the work**. Keep in mind that the work may contain inaccuracies and errors. It may be entirely unsuitable for your purpose.
- **The clearance of rights**. It is possible that the licensor might not actually own all of the rights that the licence claims to grant. If this is the case, you could inadvertently infringe the copyright of a third party — and you will bear the full responsibility of this infringement.

The licence also contains a general **disclaimer of liability**. In most cases, you cannot sue the licensor, even if you suffer injuries or financial harm due to errors, inaccuracies, or faults on the part of the licensor.

6.2.4 Licence versioning

Even if the licensor releases a new version of the licence with different terms, you can **continue your use of the work under the original licence**.

However, if you retrieve an updated copy of the work, it may then come under the terms of the new licence.

6.2.5 Choice of Law and Forum

The licence **does not specify a forum** for any lawsuits. If a legal dispute arises between you and the licensor, courts will decide upon the location for the hearing based on how strongly the legal issue connects to the jurisdictions involved.

The licence **does not specify the particular law** that a court is to apply when resolving a dispute. To interpret the licence, courts will determine the appropriate law based on how strongly the legal issue relates to each jurisdiction involved.

6.3 BSD 3-clause “New” or “Revised” License

URL: http://www.freebsd.org/copyright/licence.html
This licence is maintained by **FreeBSD / UC Berkeley**.

6.3.1 Rights and Permissions

You are free to:

- **Use and reproduce** the work for your own purposes
- **Modify** the work and adapt it into your own projects
- **Distribute** the work and share it with others

Your freedoms under this licence apply to all **copyright** that vests in the work. However, you still must ensure that you do not infringe any:

- **Trade-marks**
- **Neighbouring rights (i.e. rights in a sound recording or performance)**
- **Database rights (where applicable under European law)**
- **Moral rights**

Patents held by the contributors are also **likely implicitly licensed**, but only for the types of use implicated at the time that each contributor made his or her contribution. If the software has since changed, or if you are modifying the software, beware of other patents that you might infringe.

6.3.2 Obligations

You must:

- **Retain the original copyright notices** and disclaimers to inform downstream users of the licence conditions that apply to the original work

6.3.3 Disclaimers

The licensor makes **no guarantee** to you as to:

- **The quality of the work**. Keep in mind that the work may contain inaccuracies and errors. It may be entirely unsuitable for your purpose.
- **The clearance of rights**. It is possible that the licensor might not actually own all of the rights that the licence claims to grant. If this is the case, you could inadvertently infringe the copyright of a third party — and you will bear the full responsibility of this infringement.

The licence also contains a general **disclaimer of liability**. In most cases, you cannot sue the licensor, even if you suffer injuries or financial harm due to errors, inaccuracies, or faults on the part of the licensor.

6.3.4 Licence versioning

Even if the licensor releases a new version of the licence with different terms, you can **continue your use of the work under the original licence**.

However, if you retrieve an updated copy of the work, it may then come under the terms of the new licence.

6.3.5 Choice of Law and Forum

The licence **does not specify a forum** for any lawsuits. If a legal dispute arises between you and the licensor, courts will decide upon the location for the hearing based on how strongly the legal issue connects to the jurisdictions involved.

The licence **does not specify the particular law** that a court is to apply when resolving a dispute. To interpret the licence, courts will determine the appropriate law based on how strongly the legal issue relates to each jurisdiction involved.

6.4 Eclipse Public License v 1.0

URL: http://www.eclipse.org/legal/epl-v10.html
This licence is maintained by **Eclipse Foundation**.

6.4.1 Rights and Permissions

You are free to:

- **Use and reproduce** the work for your own purposes
- **Modify** the work and adapt it into your own projects
- **Distribute** the work and share it with others

Your freedoms under this licence apply to all **copyright** that vests in the work. However, you still must ensure that you do not infringe any:

- **Trade-marks**
- **Neighbouring rights (i.e. rights in a sound recording or performance)**
- **Database rights (where applicable under European law)**
- **Moral rights**

Patents held by the contributors are also **explicitly licensed**, but only for the types of use implicated at the time that each contributor made his or her contribution. If the software has since changed, or if you are modifying the software, beware of other patents that you might infringe.

6.4.2 Obligations

You must:

- **Retain the original copyright notices** and disclaimers to inform downstream users of the licence conditions that apply to the original work
- Provide the work in a **modifiable form** that others can easily edit and add to (e.g. provide the source code of any software program)
- **Share your modifications** with others under the exact same licence as the original work

6.4.3 Share-alike / Copyleft

You may use the work and modify it without sharing your changes, as long as this is sole for your personal use. However, to promote a community of open sharing, you **must licence your changes to others** under the same terms as the original if you:

- **Distribute** or publicly communicate your modified version to anyone else, whether for free or for profit

This copyleft obligation applies to:

- Any **adaptation or derivative work** you create based on the original

but does **NOT** apply to:

- Any **compilation** or larger project, as long as it merely includes the whole of the unchanged original work

6.4.4 Disclaimers

The licensor makes **no guarantee** to you as to:

- **The quality of the work**. Keep in mind that the work may contain inaccuracies and errors. It may be entirely unsuitable for your purpose.
- **The clearance of rights**. It is possible that the licensor might not actually own all of the rights that the licence claims to grant. If this is the case, you could inadvertently infringe the copyright of a third party — and you will bear the full responsibility of this infringement.

The licence also contains a general **disclaimer of liability**. In most cases, you cannot sue the licensor, even if you suffer injuries or financial harm due to errors, inaccuracies, or faults on the part of the licensor.

You must additionally **indemnify** the licensor for lawsuits that relate to your use of the work. That is, in certain cases, if someone else sues the licensor, it's you who must pay for all of the licensor's legal costs and any damages (even if the licensor is at fault).

6.4.5 Licence versioning

Even if the licensor releases a new version of the licence with different terms, you can **continue your use of the work under the original licence**.

However, if you retrieve an updated copy of the work, it may then come under the terms of the new licence.

6.4.6 Choice of Law and Forum

The licence **does not specify a forum** for any lawsuits. If a legal dispute arises between you and the licensor, courts will decide upon the location for the hearing based on how strongly the legal issue connects to the jurisdictions involved.

To resolve any such legal disputes, courts will interpret the licence based on the **laws in force at the specific location named in the licence**.

6.5 GNU Affero General Public License v3.0

URL: http://www.gnu.org/licences/agpl.html
Maintained by the **Free Software Foundation.**

6.5.1 Rights and Permissions

You are free to:

- **Use and reproduce** the work for your own purposes
- **Modify** the work and adapt it into your own projects
- **Distribute** the work and share it with others

Your freedoms under this licence apply to all **copyright** that vests in the work. However, you still must ensure that you do not infringe any:

- **Trade-marks**
- **Neighbouring rights (i.e. rights in a sound recording or performance)**
- **Database rights (where applicable under European law)**
- **Moral rights**

Patents held by the contributors are also **explicitly licensed**, but only for the types of use implicated at the time that each contributor made his or her contribution. If the software has since changed, or if you are modifying the software, beware of other patents that you might infringe.

6.5.2 Obligations

You must:

- **Retain the original copyright notices** and disclaimers to inform downstream users of the licence conditions that apply to the original work
- Provide the work in a **modifiable form** that others can easily edit and add to (egg. provide the source code of any software program)
- **Share your modifications** with others under the exact same licence as the original work

6.5.3 Share-alike / Copyleft

You may use the work and modify it without sharing your changes, as long as this is sole for your personal use. However, to promote a community of open sharing, you **must licence your changes to others** under the same terms as the original if you:

- **Distribute** or publicly communicate your modified version to anyone else, whether for free or for profit

- Allow users to **interact with or access** your modified version of the work, whether over the internet or through another public network

This copyleft obligation applies to:

- Any **adaptation or derivative work** you create based on the original

but does **NOT** apply to:

- Any **compilation** or larger project, as long as it merely includes the whole of the unchanged original work

6.5.4 Disclaimers

The licensor makes **no guarantee** to you as to:

- **The quality of the work**. Keep in mind that the work may contain inaccuracies and errors. It may be entirely unsuitable for your purpose.
- **The clearance of rights**. It is possible that the licensor might not actually own all of the rights that the licence claims to grant. If this is the case, you could inadvertently infringe the copyright of a third party — and you will bear the full responsibility of this infringement.

The licence also contains a general **disclaimer of liability**. In most cases, you cannot sue the licensor, even if you suffer injuries or financial harm due to errors, inaccuracies, or faults on the part of the licensor.

6.5.5 Licence versioning

Even if the licensor releases a new version of the licence with different terms, you can **continue your use of the work under the original licence**.

However, if you retrieve an updated copy of the work, it may then come under the terms of the new licence.

6.5.6 Choice of Law and Forum

The licence **does not specify a forum** for any lawsuits. If a legal dispute arises between you and the licensor, courts will decide upon the location for the hearing based on how strongly the legal issue connects to the jurisdictions involved.

The licence **does not specify the particular law** that a court is to apply when resolving a dispute. To interpret the licence, courts will determine the appropriate law based on how strongly the legal issue relates to each jurisdiction involved.

6.6 GNU General Public License v3.0

URL: http://www.gnu.org/licences/gpl-3.0-standalone.html
This licence is maintained by **Free Software Foundation**.

6.6.1 Rights and Permissions

You are free to:

- **Use and reproduce** the work for your own purposes
- **Modify** the work and adapt it into your own projects
- **Distribute** the work and share it with others

Your freedoms under this licence apply to all **copyright** that vests in the work. However, you still must ensure that you do not infringe any:

- **Trade-marks**
- **Neighbouring rights (i.e. rights in a sound recording or performance)**
- **Database rights (where applicable under European law)**
- **Moral rights**

Patents held by the contributors are also **explicitly licensed**, but only for the types of use implicated at the time that each contributor made his or her contribution. If the software has since changed, or if you are modifying the software, beware of other patents that you might infringe.

6.6.2 Obligations

You must:

- **Retain the original copyright notices** and disclaimers to inform downstream users of the licence conditions that apply to the original work
- Provide the work in a **modifiable form** that others can easily edit and add to (egg. provide the source code of any software program)
- **Share your modifications** with others under the exact same licence as the original work

6.6.3 Share-alike / Copyleft

You may use the work and modify it without sharing your changes, as long as this is sole for your personal use. However, to promote a community of open sharing, you **must licence your changes to others** under the same terms as the original if you:

- **Distribute** or publicly communicate your modified version to anyone else, whether for free or for profit

This copyleft obligation applies to:

- Any **adaptation or derivative work** you create based on the original

but does **NOT** apply to:

- Any **compilation** or larger project, as long as it merely includes the whole of the unchanged original work

6.6.4 Disclaimers

The licensor makes **no guarantee** to you as to:

- **The quality of the work**. Keep in mind that the work may contain inaccuracies and errors. It may be entirely unsuitable for your purpose.
- **The clearance of rights**. It is possible that the licensor might not actually own all of the rights that the licence claims to grant. If this is the case, you could inadvertently infringe the copyright of a third party — and you will bear the full responsibility of this infringement.

The licence also contains a general **disclaimer of liability**. In most cases, you cannot sue the licensor, even if you suffer injuries or financial harm due to errors, inaccuracies, or faults on the part of the licensor.

6.6.5 Licence versioning

Even if the licensor releases a new version of the licence with different terms, you can **continue your use of the work under the original licence**.

However, if you retrieve an updated copy of the work, it may then come under the terms of the new licence.

6.6.6 Choice of Law and Forum

The licence **does not specify a forum** for any lawsuits. If a legal dispute arises between you and the licensor, courts will decide upon the location for the hearing based on how strongly the legal issue connects to the jurisdictions involved.

The licence **does not specify the particular law** that a court is to apply when resolving a dispute. To interpret the licence, courts will determine the appropriate law based on how strongly the legal issue relates to each jurisdiction involved.

6.7 GNU General Public License v2.0 only

URL: http://www.gnu.org/licences/old-licences/gpl-2.0-standalone.html
This licence is maintained by **Free Software Foundation**.

6.7.1 Rights and Permissions

You are free to:

- **Use and reproduce** the work for your own purposes
- **Modify** the work and adapt it into your own projects
- **Distribute** the work and share it with others

Your freedoms under this licence apply to all **copyright** that vests in the work. However, you still must ensure that you do not infringe any:

- **Trade-marks**
- **Neighbouring rights (i.e. rights in a sound recording or performance)**
- **Database rights (where applicable under European law)**
- **Moral rights**

Patents held by the contributors are also **likely implicitly licensed**, but only for the types of use implicated at the time that each contributor made his or her contribution. If the software has since changed, or if you are modifying the software, beware of other patents that you might infringe.

6.7.2 Obligations

You must:

- **Retain the original copyright notices** and disclaimers to inform downstream users of the licence conditions that apply to the original work
- Provide the work in a **modifiable form** that others can easily edit and add to (egg. provide the source code of any software program)
- **Share your modifications** with others under the exact same licence as the original work

6.7.3 Share-alike / Copyleft

You may use the work and modify it without sharing your changes, as long as this is sole for your personal use. However, to promote a community of open sharing, you **must licence your changes to others** under the same terms as the original if you:

- **Distribute** or publicly communicate your modified version to anyone else, whether for free or for profit

This copyleft obligation applies to:

- Any **adaptation or derivative work** you create based on the original

but does **NOT** apply to:

- Any **compilation** or larger project, as long as it merely includes the whole of the unchanged original work

6.7.4 Disclaimers

The licensor makes **no guarantee** to you as to:

- **The quality of the work**. Keep in mind that the work may contain inaccuracies and errors. It may be entirely unsuitable for your purpose.
- **The clearance of rights**. It is possible that the licensor might not actually own all of the rights that the licence claims to grant. If this is the case, you could inadvertently infringe the copyright of a third party — and you will bear the full responsibility of this infringement.

The licence also contains a general **disclaimer of liability**. In most cases, you cannot sue the licensor, even if you suffer injuries or financial harm due to errors, inaccuracies, or faults on the part of the licensor.

6.7.5 Licence versioning

Even if the licensor releases a new version of the licence with different terms, you can **continue your use of the work under the original licence**.

However, if you retrieve an updated copy of the work, it may then come under the terms of the new licence.

6.7.6 Choice of Law and Forum

The licence **does not specify a forum** for any lawsuits. If a legal dispute arises between you and the licensor, courts will decide upon the location for the hearing based on how strongly the legal issue connects to the jurisdictions involved.

6.8 GNU Lesser General Public License v3.0

URL: http://www.gnu.org/licences/lgpl-3.0-standalone.html
This licence is maintained by **Free Software Foundation**.

6.8.1 Rights and Permissions

You are free to:

- **Use and reproduce** the work for your own purposes
- **Modify** the work and adapt it into your own projects
- **Distribute** the work and share it with others

Your freedoms under this licence apply to all **copyright** that vests in the work. However, you still must ensure that you do not infringe any:

- **Trade-marks**
- **Neighbouring rights (i.e. rights in a sound recording or performance)**
- **Database rights (where applicable under European law)**
- **Moral rights**

Patents held by the contributors are also **explicitly licensed**, but only for the types of use implicated at the time that each contributor made his or her contribution. If the software has since changed, or if you are modifying the software, beware of other patents that you might infringe.

6.8.2 Obligations

You must:

- **Retain the original copyright notices** and disclaimers to inform downstream users of the licence conditions that apply to the original work
- Provide the work in a **modifiable form** that others can easily edit and add to (egg. provide the source code of any software program)
- **Share your modifications** with others under the exact same licence as the original work

6.8.3 Share-alike / Copyleft

You may use the work and modify it without sharing your changes, as long as this is sole for your personal use. However, to promote a community of open sharing, you **must licence your changes to others** under the same terms as the original if you:

- **Distribute** or publicly communicate your modified version to anyone else, whether for free or for profit

This copyleft obligation applies to:

- Any **adaptation or derivative work** you create based on the original

but does **NOT** apply to:

- A work you create that only programmatically **links** to an original software work
- Any **compilation** or larger project, as long as it merely includes the whole of the unchanged original work

6.8.4 Disclaimers

The licensor makes **no guarantee** to you as to:

- **The quality of the work**. Keep in mind that the work may contain inaccuracies and errors. It may be entirely unsuitable for your purpose.
- **The clearance of rights**. It is possible that the licensor might not actually own all of the rights that the licence claims to grant. If this is the case, you could inadvertently infringe the copyright of a third party — and you will bear the full responsibility of this infringement.

The licence also contains a general **disclaimer of liability**. In most cases, you cannot sue the licensor, even if you suffer injuries or financial harm due to errors, inaccuracies, or faults on the part of the licensor.

6.8.5 Licence versioning

Even if the licensor releases a new version of the licence with different terms, you can **continue your use of the work under the original licence**.

However, if you retrieve an updated copy of the work, it may then come under the terms of the new licence.

6.8.6 Choice of Law and Forum

The licence **does not specify a forum** for any lawsuits. If a legal dispute arises between you and the licensor, courts will decide upon the location for the hearing based on how strongly the legal issue connects to the jurisdictions involved.

The licence **does not specify the particular law** that a court is to apply when resolving a dispute. To interpret the licence, courts will determine the appropriate law based on how strongly the legal issue relates to each jurisdiction involved.

6.9 GNU Lesser General Public License v2.1

URL: http://www.gnu.org/licences/old-licences/lgpl-2.1-standalone.html
This licence is maintained by **Free Software Foundation** for use with **software**.

6.9.1 Rights and Permissions

You are free to:

- **Use and reproduce** the work for your own purposes
- **Modify** the work and adapt it into your own projects
- **Distribute** the work and share it with others

Your freedoms under this licence apply to all **copyright** that vests in the work. However, you still must ensure that you do not infringe any:

- **Trade-marks**
- **Neighbouring rights (i.e. rights in a sound recording or performance)**
- **Database rights (where applicable under European law)**
- **Moral rights**

Patents held by the contributors are also **likely implicitly licensed**, but only for the types of use implicated at the time that each contributor made his or her contribution. If the software has since changed, or if you are modifying the software, beware of other patents that you might infringe.

6.9.2 Obligations

You must:

- **Retain the original copyright notices** and disclaimers to inform downstream users of the licence conditions that apply to the original work
- Provide the work in a **modifiable form** that others can easily edit and add to (egg. provide the source code of any software program)
- **Share your modifications** with others under the exact same licence as the original work

6.9.3 Share-alike / Copyleft

You may use the work and modify it without sharing your changes, as long as this is sole for your personal use. However, to promote a community of open sharing, you **must licence your changes to others** under the same terms as the original if you:

- **Distribute** or publicly communicate your modified version to anyone else, whether for free or for profit

This copyleft obligation applies to:

- Any **adaptation or derivative work** you create based on the original

but does **NOT** apply to:

- A work you create that only programmatically **links** to an original software work
- Any **compilation** or larger project, as long as it merely includes the whole of the unchanged original work

6.9.4 Disclaimers

The licensor makes **no guarantee** to you as to:

- **The quality of the work**. Keep in mind that the work may contain inaccuracies and errors. It may be entirely unsuitable for your purpose.
- **The clearance of rights**. It is possible that the licensor might not actually own all of the rights that the licence claims to grant. If this is the case, you could inadvertently infringe the copyright of a third party — and you will bear the full responsibility of this infringement.

The licence also contains a general **disclaimer of liability**. In most cases, you cannot sue the licensor, even if you suffer injuries or financial harm due to errors, inaccuracies, or faults on the part of the licensor.

6.9.5 Licence versioning

Even if the licensor releases a new version of the licence with different terms, you can **continue your use of the work under the original licence**.

However, if you retrieve an updated copy of the work, it may then come under the terms of the new licence.

6.9.6 Choice of Law and Forum

The licence **does not specify a forum** for any lawsuits. If a legal dispute arises between you and the licensor, courts will decide upon the location for the hearing based on how strongly the legal issue connects to the jurisdictions involved.

The licence **does not specify the particular law** that a court is to apply when resolving a dispute. To interpret the licence, courts will determine the appropriate law based on how strongly the legal issue relates to each jurisdiction involved.

6.10 MIT License / X.Org Preferred License

URL: http://www.x.org/releases/X11R7.7/doc/xorg-docs/License.html
This licence is maintained by **X.Org Foundation** for use with **software**.

6.10.1 Rights and Permissions

You are free to:

- **Use and reproduce** the work for your own purposes
- **Modify** the work and adapt it into your own projects
- **Distribute** the work and share it with others

Your freedoms under this licence apply to all **copyright** that vests in the work. However, you still must ensure that you do not infringe any:

- **Trade-marks**
- **Neighbouring rights (i.e. rights in a sound recording or performance)**
- **Database rights (where applicable under European law)**
- **Moral rights**

Patents held by the contributors are also **likely implicitly licensed**, but only for the types of use implicated at the time that each contributor made his or her contribution. If the software has since changed, or if you are modifying the software, beware of other patents that you might infringe.

6.10.2 Obligations

You must:

- **Retain the original copyright notices** and disclaimers to inform downstream users of the licence conditions that apply to the original work

6.10.3 Disclaimers

The licensor makes **no guarantee** to you as to:

- **The quality of the work**. Keep in mind that the work may contain inaccuracies and errors. It may be entirely unsuitable for your purpose.
- **The clearance of rights**. It is possible that the licensor might not actually own all of the rights that the licence claims to grant. If this is the case, you could inadvertently infringe the copyright of a third party — and you will bear the full responsibility of this infringement.

The licence also contains a general **disclaimer of liability**. In most cases, you cannot sue the licensor, even if you suffer injuries or financial harm due to errors, inaccuracies, or faults on the part of the licensor.

6.10.4 Licence versioning

Even if the licensor releases a new version of the licence with different terms, you can **continue your use of the work under the original licence**.

However, if you retrieve an updated copy of the work, it may then come under the terms of the new licence.

6.10.5 Choice of Law and Forum

The licence **does not specify a forum** for any lawsuits. If a legal dispute arises between you and the licensor, courts will decide upon the location for the hearing based on how strongly the legal issue connects to the jurisdictions involved.

The licence **does not specify the particular law** that a court is to apply when resolving a dispute. To interpret the licence, courts will determine the appropriate law based on how strongly the legal issue relates to each jurisdiction involved.

6.11 Mozilla Public License Version 2.0

URL: http://www.mozilla.org/MPL/2.0/
This licence is maintained by **Mozilla Foundation** for use with **software**.

6.11.1 Rights and Permissions

You are free to:

- **Use and reproduce** the work for your own purposes
- **Modify** the work and adapt it into your own projects
- **Distribute** the work and share it with others

Your freedoms under this licence apply to all **copyright** that vests in the work. However, you still must ensure that you do not infringe any:

- **Trade-marks**
- **Neighbouring rights (i.e. rights in a sound recording or performance)**
- **Database rights (where applicable under European law)**
- **Moral rights**

Patents held by the contributors are also **explicitly licensed**, but only for the types of use implicated at the time that each contributor made his or her contribution. If the software has since changed, or if you are modifying the software, beware of other patents that you might infringe.

6.11.2 Obligations

You must:

- **Retain the original copyright notices** and disclaimers to inform downstream users of the licence conditions that apply to the original work
- Provide the work in a **modifiable form** that others can easily edit and add to (egg. provide the source code of any software program)
- **Share your modifications** with others under the exact same licence as the original work

6.11.3 Share-alike / Copyleft

You may use the work and modify it without sharing your changes, as long as this is solely for your personal use. However, to promote a community of open sharing, you **must licence your changes to others** under the same terms as the original if you:

- **Distribute** or publicly communicate your modified version to anyone else, whether for free or for profit

This copyleft obligation applies to:

- Any **individual files** from the original that you modify (and any of your files that include portions of the original work)

but does **NOT** apply to:

- Any individual **files that you author** entirely yourself
- Any **compilation** or larger project, as long as it merely includes the whole of the unchanged original work

6.11.4 Disclaimers

The licensor makes **no guarantee** to you as to:

- **The quality of the work**. Keep in mind that the work may contain inaccuracies and errors. It may be entirely unsuitable for your purpose.
- **The clearance of rights**. It is possible that the licensor might not actually own all of the rights that the licence claims to grant. If this is the case, you could inadvertently infringe the copyright of a third party — and you will bear the full responsibility of this infringement.

The licence also contains a general **disclaimer of liability**. In most cases, you cannot sue the licensor, even if you suffer injuries or financial harm due to errors, inaccuracies, or faults on the part of the licensor.

6.11.5 Licence versioning

Even if the licensor releases a new version of the licence with different terms, you can **continue your use of the work under the original licence**.

However, if you retrieve an updated copy of the work, it may then come under the terms of the new licence.

6.11.6 Choice of Law and Forum

If a legal dispute arises between you and the licensor, it must be heard by a **court at the location of the defendant**. This can have the effect of deterring lawsuits, as it can increase the cost of a foreign would-be plaintiff filing a lawsuit.

To resolve any such legal disputes, courts will interpret the licence based on the **laws in force where the lawsuit is heard**.