"Free and Open Source Software": accompanying notes

This paper accompanies Neil Brown's presentation to Informa's IT Law Summer School 2014, at Downing College, Cambridge. It is licensed under CC BY 4.0.

Introduction

The topic of Free and Open Source Software is a very broad one, and the time allowed for the talk permits only the briefest of introductions. If you are interested in learning more about the subject, the following may be of use:

Shemtov and Walden's "*Free and Open Source Software: Policy, Law and Practice*" (OUP, 2013)

"*The International Free and Open Source Software Law Book*": http://ifosslawbook.org/

"*The International Free and Open Source Software Law Review*": http://www.ifosslr.org/ifosslr

Of course, you are also more than welcome to drop me an email, and I will try to point you in the right direction: neil@neilzone.co.uk

What is ~~Free~~ software?

Tempting though it may be to jump straight in to considering what (if anything) makes Free or Open Source software different, in terms of licensing paradigms, to any other form of software licensing, a pragmatic starting point for understanding FOSS licensing — indeed, any form of software licensing — is to spend at least some time getting to grips with the basics of the underlying subject matter.

Much has been written as to "what is software" but, for the purposes of this introduction, it is sufficient to say that software is a collection of computer programs, and that a computer program consists of a series of instructions as to what operations a computer should perform.

Like a contract, or a statute, the instructions comprising a computer program should be unambiguous and capable of producing only one meaning. If unclear, the computer program might not run, or crash when running or, arguably worse, produce an erroneous output.

Many common Free and Open Source software licences pre-suppose that software is written in source code, and, through the operation of another piece of software, is turned into something which is capable of being executed by a computer: object code, or executable code, or binary code. As not all languages follow this model, when you are advising on a software-related issue, consider finding out how the particular software in question is written and put together. However, for those that do, the model looks something like this:

Source code

Source code is the human readable form of a computer program: it is what a programmer writes when she sits down at her keyboard.

Take, for example, this simple C++ program:

```
// A simple program in C++, which outputs
“Hello, World!”
      #include<iostream>
      using namespace std;
      int main()
      {
            cout << "Hello, World!";
            return 0;
      }
```

Just like reading a foreign language, you may not know what it all means, but you can, at least, read it. With sufficient time, you could look through it line by line, and understand what it does.

You would learn, for example, that prepending a line with “//” means that the compiler (of which, more later) does not treat the line as an instruction, but as a comment — guidance for those who may subsequently read the program’s source code.

Similarly, a line beginning with “#” is an instruction to the compiler’s pre-processor; in this case, it tells the compiler to include something call “<iostream>”, which is standard C++ code

(a "library") for defining input and output stream objects.[1] It is used ("called") later in the program, with the instruction "cout".[2]

If you wish to make modifications to a program — for example, to instruct it to print "Goodbye, Earth!" rather than "Hello, World!" — the simplest way of doing so is to make a change to the source code of the program.

Compilation

When you have finished your C++ program, and included the instructions which you wish the computer to perform, you need to put that set of instructions into a form which the computer can implement. This process is known as "compilation", and uses a piece of software known as a "compiler". Each compiler works in a slightly different way, but, at a high level, the compiler works through each line of the source code, taking action based on the nature of that line, and translating the high level language (source code) into a low level language (machine instructions).

Some compilers attempt to optimise the code which they are producing, and others will expand aspects of the code with pre-defined macros or other sub-routines: whilst outside the scope of this article, copyright lawyers may find much of interest within the operation of a compiler itself. Similarly, most programs will attempt to re-use code, both to avoid duplication within a program, or to make use of something which already exists and there is considerable debate as to the copyright effects of different methods of incorporating such code libraries, or interacting with code running outside the program's memory space (whether on the same computer or on a remote computer). See further Malcolm Bain's "*Software Interactions and the GNU General Public License*",[3] and the underlying working paper "*Working Paper on the legal implication of certain forms of Software Interactions (a.k.a linking)*"[4].

---

[1] http://www.cplusplus.com/reference/iostream/

[2] http://www.cplusplus.com/reference/iostream/cout/

[3] IFOSS L. Rev, 2(2), pp 165 – 180: http://www.ifosslr.org/ifosslr/article/download/44/74

[4] http://www.ifosslr.org/public/LinkingDocument.odt

Object code

Once the compiler has finished, a copy of the program capable of being implemented by the computer is produced. This version of the program is expressed in a low-level language, and is known as object code, executable code or binary code.

This form of the program is not readily understandable by humans, and is intended to be executed by the computer. The slick interface of the finished product you see when you use your computer is thus likely to be very different to the numerous text files, and binary resources such as images, video clips or audio effects, the developer sees when she is writing that software.

Object code forms the majority of the software which consumers acquire — apps from a provider's app store, software downloaded from a developer's website, or pre-loaded on a computer, such as Microsoft's Windows or Apple's Mac's operating systems. As mentioned above, however, not all software is distributed in object form: html, the hypertext language of the web, is usually transmitted from the web server to your web browser in source code form, where it is rendered by the parser in your browser, for example, as is Javascript: knowing what your client is doing and how they have done it is important.

What is Free software

Although "FOSS" — Free and Open Source Software — is often used as a catch-all description (sometimes with an additional "L" (FLOSS, incorporating "Libre")), not everyone agrees whether "Free" software and "Open Source" software are the same thing. This section explores the notion of "Free" software, and the philosophy which underpins it, and contrasts this with the marketing-originated term of "Open Source".

Free software

A term of art rather than of law, "Free" software is generally considered to be software which meets four essential freedoms.

These freedoms are:

> Freedom 0: The freedom to run the program as you wish, for any purpose

> Freedom 1: The freedom to study how the program works, and change it so it does your computing as you wish. Access to the source code is a precondition for this
>
> Freedom 2: The freedom to redistribute copies so you can help your neighbor
>
> Freedom 3: The freedom to distribute copies of your modified versions to others. 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

(Why start at zero, rather than one? It's a programming thing: the first item in an array is at position zero...)

The focus on these freedoms is inherent from the context in which "Free" software arose: a political or social, rather than purely technological, motivation:

> "We campaign for these freedoms because everyone deserves them. With these freedoms, the users (both individually and collectively) control the program and what it does for them. When users don't control the program, we call it a "nonfree" or "proprietary" program. The nonfree program controls the users, and the developer controls the program; this makes the program an instrument of unjust power."

For more on this motivation, see the FSF's "*The Free Software Definition*": https://www.gnu.org/philosophy/free-sw.html

"Free" software thus refers to the embodiment of the four freedoms, and not to price — it is not "gratis" software. This linguistic issue can easily be a source of confusion, particularly when one might download a "free" (but not Free) application, or, increasingly, a "freemium" application, which may be neither free nor Free.

As you will have noticed, none of the freedoms preclude charging for the distribution of Free software: it is permitted, even encouraged, to charge for distributing Free software,[5] and there is no limit to the price one might charge.[6]

---

[5] https://www.gnu.org/philosophy/selling.html

[6] http://www.gnu.org/licenses/gpl-faq.html#DoesTheGPLAllowMoney

Conversely, the freedoms permit someone to give away copies of the software, should they wish — this, in itself, seems confusing to some, and, in 2006, a Trading Standards officer attempted to stop someone from giving away copies of Mozilla's web browser, Firefox, and struggled to believe Mozilla's stance that this was permitted and welcomed.[7] By giving their software away for free, the officer reportedly argued:

> "it makes it virtually impossible for [Trading Standards], from a practical point of view, to enforce UK anti-piracy legislation, as it is difficult for us to give general advice to businesses over what is/is not permitted."

As software which does not meet the four freedoms is not considered "Free" software, only some of the numerous FOSS licences are "Free" software licences. The most prominent of these are the GNU General Public Licenses (version 2.0 and 3.0), and the GNU Lesser (formerly Library) General Public License, discussed below.

Open source software

The other half of the "FOSS" moniker is "open source" software. There has been considerable debate as to whether "open source" and "Free" mean the same things. Richard Stallman, who originated the Free software perspective, considers the difference to be one based on values:

> "The two terms describe almost the same category of software, but they stand for views based on fundamentally different values. Open source is a development methodology; free software is a social movement. For the free software movement, free software is an ethical imperative, essential respect for the users' freedom. By contrast, the philosophy of open source considers issues in terms of how to make software "better"—in a practical sense only. It says that nonfree software is an inferior solution to the practical problem at hand. Most discussion of "open source" pays no attention to right and wrong, only to popularity and success ."[8]

---

[7] http://boingboing.net/2006/02/23/uk-antipiracy-office.html

[8] https://www.gnu.org/philosophy/open-source-misses-the-point.html

Free software, he argues, respects freedom, whilst open source software focuses on the technical merits and business advantages of source code availability and permission to modify, to the detriment of this focus on freedom.

The Open Source Initiative, on the other hand, argues that there is no difference:

> ""Free software" and "open source software" are two terms for the same thing: software released under licenses that guarantee a certain, specific set of freedoms."[9]

Indeed, the OSI once described itself as "a marketing program for Free software":

> "It's a pitch for "free software" on solid pragmatic grounds rather than ideological tub-thumping. The winning substance has not changed, the losing attitude and symbolism have."[10]

As with the four freedoms of Free software, "open source" software is generally considered to be software licensed under terms which meet the 10-point Open Source Definition, which is set out in full at Annex 1. The Definition was originally drafted by Bruce Perens in 1997, in the context of guidelines for the operation of the Debian software project,[11] and consists primarily of the Debian Free Software Guidelines[12] minus the Debian-specific references.

Like Free software, for a licence to meet the Definition, it must permit the selling of covered software, and must not prevent the redistribution of source code.

The distinction in practice

For most lawyers, the main distinction is one of understanding the community in which a client operates, and the likely reaction

---

[9] http://opensource.org/faq#free-software

[10] From the Open Source Initiative's web page from December 2002: http://web.archive.org/web/20021217003716/http://www.opensource.org/advocacy/faq.html

[11] https://www.debian.org/

[12] https://www.debian.org/social_contract

to referring to any given project as "Free" or "open source" — whilst, for the purposes of a legal analysis, the title given to any particular licence is largely irrelevant, from a risk perspective, adhering to the relevant community norms, including naming, may be advantageous.

Spectrum of FOSS licences

There are many FOSS licences. The OSI has stated that it has received "many hundreds" of licences for consideration as official Open Source licences,[13] and, as of 2012, has approved 68.[14] One company, which provides FOSS compliance support activities, stated that it has over 1,000 FOSS-type licences in its database.[15]

In practice, there is a core of licences which are more common than others, including the key licences of the Free software movement — the GNU General Public Licenses — the MIT and BSD licences, the Apache License 2.0 and the Mozilla Public License version 1.1.[16]

Consider, if you will, a spectrum: at one end — let's say, on the left — is the notion of complete lack of encumbrance. This is the public domain. Something which is in the public domain is free from restrictions, and can be used as, when and how you wish. At the other end of the spectrum — on the right — is a work which is subject to the restrictions of copyright. Without your permission, or without the satisfaction of a statutory exemption (or, perhaps, in the US, a "fair use"[17]), someone may not perform an act restricted by copyright. To do so would infringe your copyright.

The gap in between — between the freedom of the public domain and the almost-absolute restrictions of copyright — is where copyright licences exist. This is true whether the licences are

---

[13] http://opensource.org/proliferation

[14] http://www.openlogic.com/wazi/bid/187971/Are-68-Open-Source-Licenses-Enough-Or-Too-Many

[15] OpenLogic: http://www.openlogic.com/wazi/bid/187971/Are-68-Open-Source-Licenses-Enough-Or-Too-Many

[16] See http://www.blackducksoftware.com/resources/data/top-20-open-source-licenses

[17] http://fairuse.stanford.edu/overview/fair-use/

"Free" licences or otherwise, as a licence grants permission to do something which is otherwise prohibited.

Introduction

This section will introduce you to various Free and Open Source licences, as markers of various places along this spectrum. Some are closer to the public domain, whereas, conceptually, some are closer to the restrictions of copyright.

As will be discussed, "Free" software licences are the more restrictive, in that, to achieve their purpose of preserving freedom, they need to impose limitations on the restrictions which a developer could otherwise impose on the codebase.

We will consider seven licences:

- BSD licenses
- CDDL (Common Development and Distribution License)
- MPL version 2 (Mozilla Public License)
- APL version 2 (Apache Public License)
- LGPL (Lesser General Public License)
- GPL (General Public License)
- AGPL (Affero General Public License)

BSD licences

"BSD" is the name given to a family of similar, permissive, licences, from the University of California, in Berkeley.

Because there are multiple licences which could be considered to be "the" BSD licence, good practice demands that you specify which version of the licence you wish to apply to the work in question. If you mean the three-clause BSD licence, state this, even if this would seem to be the current mainstream interpretation.

The BSD 3-clause licence[18] is at the left hand side of our spectrum: it is very broad and permissive. It permits redistribution in source and binary form, provided that the source code contains the included copyright notice and the BSD 3-clause licence text, and

[18] http://opensource.org/licenses/BSD-3-Clause

permits distribution of the binary on largely the same basis, including this information in accompanying documentation. It precludes the use of the name of the copyright holder, or the contributors, as endorsements of the distributed software, without their permission. In total, the BSD 3-clause licence is just 224 words.

A second version, the BSD 2-clause, is substantially the same, but removes the third clause expressly prohibiting endorsement.[19] This 2-clause form of the licence is similar to the MIT licence.[20]

Both 2- and 3-clause BSD licences are Free software licences as well as Open Source licences, and are considered GPL-compatible.[21]

No form of the BSD licence expressly references patent licences; some argue that this means that there is no such licence, whereas others argue that the nature of the licence is such that a patent licence is implied.

The original BSD licence ("BSD 4-clause") included an additional, "obnoxious",[22] fourth clause, at what was then section 3, which read:

> "3. All advertising materials mentioning features or use of this software must display the following acknowledgement:
>
> This product includes software developed by the University of California, Berkeley and its contributors. "[23]

Whilst this form of attribution may well have been a reasonable demand in respect of software which was, in fact, developed at Berkeley, the inclusion of the specific reference to Berkeley in the licence precluded the licence's use in respect of any non-Berkeley software. What would happen in this case is that the developer, happy with the broader principles behind this licence, would create their own version, replacing "University of California"

---

[19] http://opensource.org/licenses/BSD-2-Clause

[20] http://opensource.org/licenses/MIT

[21] https://www.gnu.org/licenses/license-list.html#ModifiedBSD

[22] https://www.gnu.org/philosophy/bsd.html

[23] https://www.openhub.net/licenses/bsd

with their own attribution. With each modified version requiring a different attribution line in advertising material, the practical difficulties such an approach caused were substantial.

This fourth clause was officially removed by the University of California in 1999.[24] With this extra clause present, the BSD 4-clause licence is a Free software licence and an Open Source licence, but is not GPL-compliant.[25]

CDDL: Common Development and Distribution License

The Common Development and Distribution License[26] — the CDDL — is a permissive licence, but further to the right of the spectrum than the BSD 3-clause.

Akin to the BSD 3-clause licence, it grants to a recipient a broad range of permissions, including to:

> "use, reproduce, modify, display, perform, sublicense and distribute the Original Software (or portions thereof), with or without Modifications, and/or as part of a Larger Work"

It permits onwards distribution of the covered code in executable (binary) form, provided that the source code is also made available under the terms of the CDDL.[27]

If you make modifications to CDDL code, these modifications are "governed by the terms of the licence", such that, if you choose to distribute your modified version, you must make the accompanying modified source code available.[28] It does not, however, compel you to make your modifications available. If you use your own modified version without distributing it, there is no requirement to make either source or binary versions available.

---

[24] http://bsd-beta.slashdot.org/story/99/09/02/189210/berkeley-removes-advertising-clause

[25] https://www.gnu.org/licenses/license-list.html#OriginalBSD

[26] http://opensource.org/licenses/CDDL-1.0

[27] Clause 3.1

[28] Clause 3.2

Unlike the BSD 3-clause licence, it incorporates patent licences from both the original developer and from subsequent contributors.[29] This grant extends only to the "Original Software", and would to preclude the original licensor from bringing patent infringement action in respect of a combination claim — the combination of the Original Software with other software, or as part of a wider system.

Similarly, it lacks an express grant of patent licence from a distributor; a distributor is not a "Contributor" for the purposes of this licence, since, to be a Contributor, one must have made modifications to the original covered code, or incorporated that code in new files. However, at clause 3.4, it precludes a distributor from imposing:

> "any terms on any Covered Software in Source Code form that alters or restricts the applicable version of this License or the recipients rights hereunder".

This could be read as precluding the enforcement of a patent, but, since the preclusion is limited the imposition of terms on software "in Source Code form", it does not expressly cover infringements from running a compiled version of the distributed code. One would need to fall back on a claim of implied licence.

The CDDL contains a termination provision where a recipient asserts a patent infringement claim against the Initial Developer or a Contributor in respect of the received software. The recipient litigant has 60 days from the point of notification to withdraw the claim, else the recipient's rights under the CDDL are terminated.

The CDDL is both a Free software and Open Source licence, but is not considered GPL-compatible, on the basis of its patent retaliation clause, and its "Required Notices" clause.[30]

MPL: the Mozilla Public Licenses

There are two versions of the Mozilla Public License: the current version, version 2.0, and the former, now deprecated, version, version 1.1. Check the licensing details carefully to ensure that you refer to the correct version of the licence. MPL 2.0 was

---

[29] Clauses 2.1(b) and 2.2(b) respectively.

[30] http://lists.gpl-violations.org/pipermail/legal/2007-July/001065.html

published in early 2012.[31] Mozilla provides a comparison of the old and new versions.[32]

MPL 2.0 is both a Free software and Open Source licence, and is considered GPL-compatible.[33]

Like the CDDL, the MPL 2.0 grants a wide-ranging copyright licence, and a similarly-constrained patent licence. Indeed, clause 2.3(b) specifically provides that no patent licence is granted in respect of "the combination of [a Contributor's] Contributions with other software".

All distributions of MPL 2.0 code, including modifications, must be licensed under the MPL 2.0, to preserve rights downstream, and the source code of code distributed in executable form must be made available.[34] "Modifications" are defined as changes to the content of a covered source code file, or the inclusion of covered code in any new file. This gives the MPL 2.0 a copyleft effect, at a discrete file level.[35]

Distributors are obliged to inform recipients of the availability of the source code, including how they can obtain the source in a timely manner, at a charge "no more than the cost of distribution to the recipient".[36]

As such, whilst a distributor is entitled to charge what he wishes for distribution of the executable form (or for warranty, support, indemnity or liability obligations[37]), he is not permitted to charge in respect of source code, but can recover costs.

The MPL 2.0 contains, at clause 5.2, a patent retaliation clause similar to that of the CDDL, but stricter: if a recipient initiates patent infringement litigation against any entity in respect of covered code, all rights granted to that recipient in respect of that code terminate automatically, without notice.

---

[31] https://www.fsf.org/blogs/licensing/mpl-2.0-release

[32] https://www.mozilla.org/MPL/2.0/differences.html

[33] http://www.gnu.org/licenses/license-list.html#MPL-2.0

[34] Clause 3.1

[35] https://www.mozilla.org/MPL/2.0/FAQ.html#what-is-the-mpl

[36] Clause 3.2

[37] Clause 3.5

A brief digression: the second version of this licence[38] — the current version — is also notable for its gorgeous typography, making it easy to read, thanks to then-Mozilla attorney, Luis Villa, drawing on Matthew Butterick's excellent "*Typography for Lawyers*"[39]. The disclaimer (6) and limitation of liability (7) sections demonstrate particularly good typography, using a yellow background to highlight their importance, rather than the more common (American?) style of putting these clauses in capital letters, which makes them largely unreadable.

APL version 2 (Apache Public License, version 2.0

There are three versions of the Apache Public License: 1, 1.1 and 2. Version 2.0 is the current version, and was approved by the Apache Software Foundation — perhaps best known for its eponymous web server — in 2004.[40]

The APL version 2.0 is both a Free software and Open Source licence; it is incompatible with GPLv2, but considered compatible with GPLv3.[41] It is, broadly, a permissive licence, falling somewhere in the middle of our licensing spectrum.

Permission is granted to modify the covered code, and to distribute these modifications, provided that certain notification requirements are met.[42] The APL 2.0 permits modifications to be licensed under terms other than the APL 2.0:

> "You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License."[43]

---

[38] https://www.mozilla.org/MPL/2.0/

[39] http://typographyforlawyers.com/

[40] http://www.apache.org/licenses/

[41] http://www.apache.org/licenses/GPL-compatibility.html

[42] Clause 4

[43] Clause 4

The APL 2.0 contains a patent licence, again limited to use of the covered work rather than any combination claims. This patent licence terminates automatically if the recipient institutes patent litigation against anyone, alleging that the covered work constitutes patent infringement.[44] Unlike the MPL 2.0, only the patent licence is terminated under APL 2.0; copyright licences remain in effect.

Unlike the previous version (version 1.1),[45] which required that the entire licence be reproduced in each source code file[46], version 2.0 requires only that each source file contains "attribution notices", referring to, but not incorporating the full text of, the licence.

LGPL (Lesser General Public License)

The LGPL is the Lesser (formerly Library) version of the General Public License. There are two common versions: 2.1, and 3.0.

The LGPL was drafted with a particular function in mind: to enable the combination of Free software with non-Free software, to allow a Free software library to replace a non-Free alternative. For example, it permits the developer of a non-Free program to use a Free library to perform a particular function, rather than a non-Free equivalent. In this way, the political goal of Free software — eradicating non-Free software — may be attained gradually.

Where the Free library provides enhanced functionality, or is more than a simple drop-in replacement for the non-Free alternative, the Free Software Foundation suggests using the GNU GPL (below) instead.[47]

Version 3 of the LGPL acts as a modifier to GPLv3. The main effect of this modification is set out in section 3, which provides that:

> "The object code form of an Application may incorporate material from a header file that is part of

---

[44] Clause 3

[45] http://www.apache.org/licenses/LICENSE-1.1

[46] Apache Software License, version 1.1, at clause 1

[47] https://www.gnu.org/licenses/why-not-lgpl.html

> the Library. You may convey such object code under terms of your choice … if the incorporated material is not limited to numerical parameters, data structure layouts and accessors, or small macros, inline functions and templates (ten or fewer lines in length) …"

provided that the distributor gives prominent notice of the use of the library, and accompanies the referring object code with a copy of GNU GPLv3 and LGPLv3.

The practical effect of this is that, provided that she complies with the notification and licence distribution obligations, a developer can link to an LGPLv3'd library without needing to license her referring application under GNU GPLv3; she can use a licence of her own choice. A similar permission exists in respect of the combination of LGPLv3'd code in non-LGPLv3'd libraries.[48]

LPGL 3.0 is not compatible with GPLv2 by itself. However, if the version of GPLv2 covering the code in question contains the "or any later version" clause, a recipient may choose to change the licence of that code from GPLv2 to GPLv3, thereby achieving compatibility.[49]

GPL (General Public License)

The General Public License — or, at least, one of the two common variants — is perhaps the best known of the FOSS licences. The third version of the GPL was released in 2007, following an extensive consultation; it supersedes GPLv2 as the current version of the licence. GPLv2, however, remains highly relevant, and a solid understanding of both licences, and their different regimes, is worthwhile.

At a high level, both licences are considered "strong" copyleft: that is to say, modified version of GPLv3'd code must be licensed under GPLv3 terms.[50] Similar provision exists in respect of GPLv2.[51]

---

[48] Section 5

[49] https://www.gnu.org/licenses/license-list.html#LGPL

[50] GNU General Public License version 3, at section 5(c)

[51] GNU General Public License version 2, at section 2(b)

Neither version of the licence requires modifications to be made available where those modifications are just run on the modifying party's own computers but, if the modifications are distributed (or conveyed, in GPLv3 parlance) in binary form, they must be made available in source code form too.[52] The manner in which source code must be made available varies between the two licences.

For example, whilst expressly permitted by GPLv3,[53] a distributor may be concerned about relying on hosted distribution of source code in respect of GNU GPL 2.0 code since, on a strict reading of the requirement of the "written offer" distribution mechanism, such an approach is non-compliant: the distribution is not made "on a medium".[54] However, in practice, it is perhaps unlikely that, in itself, making the source code available in this form would be considered an infringement.[55]

Alongside the distribution of the relevant source code, GPLv3 requires that, in respect of what is deemed a "User Product", "Installation Information" must be provided.[56] This is defined as:

> "any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made."

This can be contrasted with the obligation under GPLv2, where the source code distribution requirement extends to:

> "all the source code for all modules it contains, plus any associated interface definition files, plus the

---

[52] GPLv3 at section 6; GPLv3 at section 3

[53] GPLv3, section 6(d)

[54] GPLv2, section 3(b)

[55] "*GNU GPL 2.0 and 3.0: obligations to include licence text, and provide source code*" IFOSS L. Rev., 2(1), pp 7 – 12

[56] GPLv3, section 6, under the bullet points

> scripts used to control compilation and installation of the executable"[57]

The reason for the change — and a reason why some may be concerned about GPLv3 — is to avoid what is known colloquially as "Tivoisation". This is the situation in which a manufacturer produces a device containing Free software, but locks down that device in such a way that it will not run modified versions. A recipient may be given the source code, but is precluded from exercising the freedoms intended to be granted by the GPL. This may take the form, for example, of requiring software to be signed by a particular key, such that unsigned software will not run. GPLv3 attempts to prevent this situation by including within the distribution requirements all information needed to install and execute modified versions of the covered code.

This requirement only exists where it is technically possible for the manufacturer to install modified code: if, for example, the software has been installed on a read-only medium, such that, one written, it cannot be over-written, even by the device manufacturer, the requirement falls away.[58]

GPLv3 introduces patent licensing language not included in the previous version, requiring that, where someone conveys a covered work "knowingly relying" on a patent licence which would preclude others from copying the covered work, the conveying party must either rectify the situation, or "deprive [himself] of the benefit of the patent license for this particular work".[59]

This is a narrower form of patent retaliation clause than appears in some other licences; the rationale for this was that some patent retaliation clauses "promise more to users than they can really deliver", and that the clauses included within GPLv3 "may actually have meaningful effect".[60]

---

[57] GPLv2, section 3, under the bullet points

[58] GPLv3, section 6, third to last paragraph

[59] GPLv3, section 11, fifth paragraph

[60] Eben Moglen, speaking at the open session of the first international GPLv3 conference, January 2006: http://www.ifso.ie/documents/gplv3-launch-2006-01-16.html#em-patent-retaliation

Richard Stallman summarised the intended harm to be tackled in the following terms:

> "it's meant to protect against one particular kind of abuse on the part of server operators where they make an improvement, which they're free to do, and run it on their servers and they don't release their source code and if the code does not have the Affero clause on it then they don't have to release the source code, and then you decide that you are going to implement a similar improvement and then they sue you for patent infringement."[61]

GPLv2 and v3 are both Free software and Open Source software and, *de facto*, both are GPL-compliant. However, GPLv2 is not compatible with GPLv3, as some of the requirements within GPLv3 — such as the provision of installation information, above — do not exist within GPLv2.[62] However, if the code in question is covered by GPLv2 with the "any later version" language,[63] a recipient can choose to follow the terms of GPLv3 instead of GPLv2, removing this incompatibility.

AGPL (Affero General Public License)

As with the LGPL, the Affero GPL[64] tweaks the GPL to cover a particular situation, which Stallman put in the following terms:

> "One reason you should not use web applications to do your computing is that you lose control. ... "It's just as bad as using a proprietary program. Do your own computing on your own computer with your copy of a freedom-respecting program. If you use a proprietary program or somebody else's web server, you're defenceless. You're putty in the hands of whoever developed that software."[65]

---

[61] Richard Stallman, speaking in Turin, March 2006: http://fsfe.org/campaigns/gplv3/torino-rms-transcript.en.html#limited-retal

[62] http://www.gnu.org/licenses/gpl-faq.html#v2v3Compatibility

[63] GPLv3, section 9

[64] http://www.gnu.org/licenses/agpl-3.0.html

[65] Richard Stallman, "The Guardian", 29th September 2008: http://www.theguardian.com/technology/2008/sep/29/cloud.computing.richard.stallman

"Cloud" computing, he notes, equates with "Service as a Software Substitute", and "is another way to let someone else have power over your computing."[66]

As you may recall from the preceding section, the obligation within the GPL to distribute source code arises in the event of distribution (GPLv2) or conveyance (GPLv3) of covered code.

"Distribution" is not defined within GPLv2, but the license provides that "[t]he act of running the Program is not restricted".[67]

"Conveyance" is defined by GPLv3 as

> "To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying."[68]

"Propagation" is itself defined as:

> "To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well."

It is implicit within GPLv2, but expressly called out within GPLv3, that neither distribution nor conveyance encompasses "[m]ere interaction with a user through a computer network, with no transfer of a copy". Thus, running a GPL'd web server does not trigger an obligation to make available the source code to that web server. Conversely, making a binary of that web server available for download is caught. Similarly, some parts of a web page are downloaded by a web browser and executed locally, such as a Javascript; as the Javascript code is copied to the user's

---

[66] https://www.gnu.org/philosophy/who-does-that-server-really-serve.html

[67] GPLv2, section 0

[68] GPLv3, section 0

machine, this exclusion falls away, leaving the code subject to the general provisions of the relevant licence.

With the resurgence of distributed computing, particularly under the moniker of "cloud" computing, users are increasingly turning to interactions with software running on third party servers. Whilst the users make use of this software, and are reliant on it for the processing operations in respect of their data, they do not have access to the source code, and thus are unable to avail themselves of the freedoms which Free software aims to protect.

The AGPLv3 is one of a number of licences which attempts to bring network interaction within the ambit of source code distribution, and other, obligations. It does so in the following terms:

> "Notwithstanding any other provision of this License, if you modify the Program, your modified version must prominently offer all users interacting with it remotely through a computer network (if your version supports such interaction) an opportunity to receive the Corresponding Source of your version by providing access to the Corresponding Source from a network server at no charge, through some standard or customary means of facilitating copying of software."

Apple's Public Source License version 2.0[69] accomplishes a similar objective through its definition of "externally deploy", which is akin to GPLv3's notion of "conveyance":

> "to use Covered Code, alone or as part of a Larger Work, in any way to provide a service, including but not limited to delivery of content, through electronic communication with a client other than You."[70]

The inclusion of network interaction as a trigger for source distribution places the AGPL at the far right of the FOSS spectrum: in my view, this is as close as any FOSS license gets to the restrictions of a "proprietary" software licence.

---

[69] http://www.opensource.apple.com/license/apsl/

[70] Apple Public Source License, section 1.4(b)

Whether this is considered desirable will, inevitably, depend on your client's stance: if your client is looking to produce a new, competitor-beating cloud service stack, releasing it under AGPL terms may engender community contributions whilst preventing a third party from taking it and benefiting from it without releasing their changes. Conversely, someone looking to make a revenue stream from licensing such software is unlikely to find the AGPL an attractive option, and may consider than any inclusion of AGPL'd code is highly damaging to their business model.

The AGPL is both a Free software and Open Source licence, but is not GPLv2-compliant because of the extra obligation in respect of network interaction.

Warning: the AGPL was initially drafted by a different body to the GPL, and thus was not part of the "official" suite of GPL documents; it was not drafted or officially adopted by the Free Software Foundation. However, when GNU GPLv3 was released, the FSF released its own version of the AGPL, designed to accomplish similar goals as to the original AGPL. If you should be asked to advise on an "AGPL", make sure you are looking at the correct one. This section looked solely at the GNU Affero GPL, released in November 2007.[71]

Copyleft

A number of the licences discussed above were described as being "copyleft" licences.

The term "copyleft" is a play on "copyright", and references the use, perhaps even subversion, of the copyright system to secure the opposite end: the idea of preserving freedoms and users' rights, rather than restricting them.

The notion which underpins "copyleft" is perhaps described most aptly as "share, and share alike". It permits modifications to be made to covered code, but requires that, where this code is distributed or conveyed, the modifications are also made available in source form, under the same licensing terms, so that a downstream recipient's freedoms are preserved.

---

[71] http://www.gnu.org/licenses/agpl-3.0.html

Some talk of the different "strengths" of copyleft inherent in a particular licence. The MPLv2, for example, is described by Mozilla as having a "file-based" approach to copyleft, meaning that only changes to covered source files, or the inclusion of covered code in new files, trigger copyleft obligations. The inclusion of entirely new files, which do not incorporate MPLv2'd code, does not render those new files subject to the MPLv2.

The LGPLv3 has a slightly weak copyleft approach too, in that one is permitted to include an LGPL'd library within a wider program and not license that entire program under GPLv3 or LGPLv3 terms. Conversely, a common interpretation of the GPLv3 is that the linking of a new program with a GPLv3'd program renders that new program subject to GPLv3 licensing terms — this would be a stronger form of copyleft. The AGPLv3's approach of including within the ambit of distribution obligations mere network interaction makes it the strongest copyleft of those licences discussed here.

One occasionally hears copyleft licences being described as "viral"; this terms has gained some prominence after being used by senior Microsoft executive Craig Mundie.[72] Mundie commented that:

> "[t]his viral aspect of the GPL poses a threat to the intellectual property of any organization making use of it. It also fundamentally undermines the independent commercial software sector because it effectively makes it impossible to distribute software on a basis where recipients pay for the product rather than just the cost of distribution."

Describing copyleft in this manner is a best disingenuous, at worst untruthful and misleading: unlike a virus, a developer has a choice as to whether they wish to adopt a copyleft approach or not. No-one is forced to license their code under copyleft terms, nor does copyleft suddenly attract to someone's code without their knowledge or permission: it is always a developer's choice as to whether they wish to make use of existing copyleft software.

In any case, none of the mainstream licences require that modifications are made available; rather, the licenses require that, where a developer chooses to make their modifications available,

---

[72] http://www.microsoft.com/en-us/news/exec/craig/05-03sharedsource.aspx

they must also provide the source code; the control is with the developer as to whether he wishes to make his modified code available in the first place.

FOSS in the cloud

See discussion above in respect of "Affero GPL".

Is a FOSS licence actually a contract?

There is a body of opinion which holds that certain FOSS licences, particularly, but not exclusively, the GPLs, go beyond the remit of mere copyright licences, and are actually contractual in nature. This is largely predicated on the principle that the notion of copyleft — that the source code of distributed modifications must be made available — goes beyond merely granting permission, and thus exceeds the scope of a licence. The document must thus be a contract, embodying a licence but containing more than just a licence.

A contrary school of thought is that the document is a licence, with the *quid pro quo* being the conditions which a licensee must fulfil in order to benefit from the licence: the licensee is permitted to perform acts which would otherwise be restricted, provided that the licensee meets the other criteria within the licence.

There are two reasons why this might be important to you: first, the remedies available for breach of contract and infringement of copyright are not identical, and second, the potential scope of a claim, in terms of those who could be sued, is different.

From an English law perspective, the fundamental question is whether the prerequisites of a contract are present. For a contract to exist, one must have offer, acceptance, consideration, and formative requirements such as capacity, the intention to create legal relations, and a lack of vitiating factors. Without these, one has no contract or, without the ability to prove these, no enforceable contract.

Sometimes, the language of the licences does not make such an analysis easy. Take, for example, the notion of a bare licence: there is no requirement of acceptance, since it is simply a grant of permission. If you do not comply with its terms, you infringe whatever the protected right is, be it an act restricted by

copyright or a right to exclude enshrined within a form of trespass. Even if you do not "accept" something, you need to operate within its limitations if you are to avoid infringing. Conversely, for a contract, there must be an acceptance of the offered terms. If there is not, there is no contract, so a failure to comply with its purported terms cannot be a breach of contract.

The GPLv3, however, conflates the notions of licence and contract, at section 9:

> "You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so."

It talks in terms of "accepting this License", and, indeed, acceptance by conduct: "by modifying or propagating a covered work, you indicate your acceptance of this License". However, it also uses the language of exclusion via copyright, and thus licensing: "nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright…"

As requirements for contract vary between jurisdictions, a clear analysis will be required of the document in question to understand its foundation. See, for example, Mark Henley's analysis of the US case, *Jacobsen v. Katzer*[73] in respect of one of the Artistic Licenses[74].

Remedies

Typically, breach of contract and infringement of copyright attract different remedies.

---

[73] Mark Henley "*Jacobsen v Katzer and Kamind Associates — an English legal perspective*", IFOSS L. Rev., 1(1), pp 41 – 44

[74] http://opensource.org/licenses/artistic-license

Generally, damages are considered as the appropriate remedy for a breach of contract case, with the object of putting the parties in the position in which they would have been had the contract been performed,[75] subject to rules around mitigation of loss, and the principle of remoteness.[76]

Equitable remedies, such as a mandatory injunction (such as specific performance) or a prohibiting injunction (such as an order not to continue distributing GPLv3'd code in binary form only) are less common.

In respect of actions for infringement of copyright, injunctive relief is common where a claimant can demonstrate to the court that damages would be an inadequate remedy.[77] Where a claimant's loss could be remediated in damages, an injunction is unlikely. Since the cornerstone of a copyleft licence is that notion of "share, and share alike", it is not obvious that damages would be an adequate remedy; if the ill to be stopped is the promulgation of non-Free software, giving someone money but continuing to promulgate non-Free software whilst relying on that person's Free software is unlikely to be sufficient.

s97A, Copyright, Designs and Patents Act 1988, contains a specific provision permitting injunctive relief against hosting providers:

> "The High Court (in Scotland, the Court of Session) shall have power to grant an injunction against a service provider, where that service provider has actual knowledge of another person using their service to infringe copyright."

Whether a copyright holder of a FOSS-licensed work seeking to enforce its copyright would consider using a controversial mechanism such as this is questionable, but this is a potential avenue within the statute books.

---

[75] *Robinson v Harman* [1848] 18 LJ Ex 202

[76] *Hadley v Baxendale* [1854] 9 Exch. 341

[77] US centric, but see Doug Rendleman's, "*The Inadequate Remedy at Law Prerequisite for an Injunction*" (1981). Faculty Publications. Paper 886. http://scholarship.law.wm.edu/facpubs/886

It is often posited that a mandatory injunction could be used to compel someone to release their source code, and that, as such, copyleft licences are "viral" or pose a risk to businesses. To date, there has been no request for a ruling on this point, and it is not clear that, even if one were to be sought, it would succeed; at most, the court could order that, if the defendant wished to continue to distribute the work based on a copyleft work, it must comply with that copyleft work's licensing conditions.

These conditions are likely to include promulgation of the complete corresponding source code, but the onus is on the defendant to determine whether or not he wishes to continue to promulgate the work or not — it would likely be open for the defendant to stop distributing the work. At this point, there may be an issue of past infringement in respect of which remedy may be required but, since damage has already occurred, it is unlikely that an injunction would be ordered in respect of this historic infringement.

Relationships

Aside from the remedy which might be sought, the nature of the claim may also have a bearing on the scope of potential defendants.

An action for breach of contract requires that the claim is brought against another party to the contract. If Person A were to enter into a contract with Person B, which permitted B to use the software only in a certain manner, A would be entitled to sue B in the event that B used the software in a different manner.

However, if B gave the software to Person C, and C used the software in a manner which did not meet the terms of the contract, A has no obvious claim against C for breach of contract, since C was not a party to the contract between A and B. A may have a claim against B, if the contract imposed an obligation on B to flow-down the contract terms to any third parties to which it gave the software and take responsibility for any breach by a downstream recipient of those terms, but this would inevitably depend on the construction of the contract; in any case, it does not establish a cause of action against C directly.

Conversely, no nexus is required in respect of an action for copyright infringement: a party which performs an act restricted

by copyright without statutory permission or a licence from the copyright owner infringes that owner's copyright. It does not matter whether the purportedly infringing party had seen a licence document or not since, without a licence, the party had no permission to perform the act in question. In this sense, the scope of potential claims available under copyright law is likely to be broader, permitting action to be taken against third party infringers with no direct relationship to the copyright owner.[78]

Conversely, interpreting a FOSS document as a licence may preclude action being taking by a third party recipient — such as someone who buys a router containing FOSS code — against the distributor of such a router. Since action for copyright infringement can only be brought by a copyright owner or exclusive licensee,[79] a consumer who has bought a router which contains FOSS code but finds that the source code is not available cannot, on a copyright analysis, sue the distributor to obtain that source code. In such a case, the disappointed party would need to engage the copyright holder, and persuade them to take action.

The disappointed party might have a cause of action under consumer protection law — returning the goods on the basis that they are unfit for purpose, since they fail to include an element which they could reasonably expect them to include — but, unless done in sufficient quantities, this is unlikely to trouble many retailers into compliance.

Had the document been a contract, it might have been possible for the disappointed party to assert that it was a beneficiary of the contract by virtue of the Contracts (Rights of Third Parties) Act 1999, although, to date, no such claim has been tested.

Enforcement

For a detailed analysis of GPL enforcement and compliance activities, see Shemtov and Walden's "*Free and Open Source Software: Law, Policy and Practice*" at chapter 8.

---

[78] There may be a direct relationship in reality since section 10, GPLv3, for example, provides that "the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License".

[79] s96, Copyright, Designs and Patents Act 1988

The "Enforcement case studies" section of Bradley Kuhn's (and others) draft book "*Copyleft and the GNU General Public License: A Comprehensive Tutorial*" may also be useful.[80]

Why pick a FOSS licence for own software?

Why might someone choose to license their own software under FOSS terms? To copyright lawyers used to the idea of maximising restrictions, and licensing out on the most prohibitive terms possible, the idea that someone might create something and then give it away might seem alien.

As discussed in the introduction to Free software, above, for some, ensuring software freedom is a political movement and statement — that non-Free software is an evil which should be abolished. Some, perhaps many, Free software developers write software under FOSS licence terms to give effect to this belief, or because they seen no value in restricting others' access to their work. I, for example, have licensed this document under CC BY 4.0 (attribution only) terms, to remove unnecessary copyright restrictions which would prevent modification or simple redistribution of this document.

Conversely, others may choose to license their own software under FOSS terms for commercial or business, rather than political or ideological, reasons. Three of these are explored briefly below, as food for thought.

To build a community

You may have a great piece of software, but not have the resource, or other capability, to take it where you want it to go on your own — you need the involvement of others. You could take a more traditional route, and hire (and pay) third party developers. Alternatively, you could make your code base available to anyone with the skill and inclination to work on it, and benefit from the changes which they might wish to make.

There would be no guarantee, of course, that anyone did want to work on your project, and it is unlikely that, in the early days, at least, it would be the same kind of "work" that you might direct

---

[80] Currently available at http://ebb.org/bkuhn/writings/comprehensive-gpl-guide.pdf

of a hired developer: rather than someone building what you want, they are more likely to make changes which do something which they want. You may end up with a richer, more capable piece of software, but it might not be what you had originally envisaged.

To undercut a rival's investment

Consider the situation in which you have spent months developing a new piece of software to do something amazing — something for which there is nothing else quite like it on the market. And, a couple of months before you are ready to launch, someone else puts something substantially similar on the market. They win customers immediately, eating heavily into your projected market space.

You have choices:

You could start selling your software immediately, before it is quite ready, and try to win a chunk of the market, but at the risk that your software fails to perform adequately.

You could wait until your software is ready, but then enter a market where someone already has the benefit of the first-to-market advantage — you might be able to win customers who have not already bought the other company's solution, but the size of the market is much smaller; you are unlikely to attract customers who have just bought the other company's solution unless yours is much better, or theirs is failing.

You could try to undercut your rival's investment, and regain the focus. One mechanism to do this might be to release your software as FOSS, and, additionally, not charge for it — assuming that your software is comparable and not materially worse, new customers have the choice of either buying your rival's software, or taking yours without charge. By making your software available on both gratis and FOSS terms, someone who has already invested in your rival's solution may take the opportunity of downloading yours and experimenting with it, where, had they had to pay, they may not have done so.

Likewise, you could offer the core of your software on FOSS terms, and charge for proprietary modules and plug-ins; this may be particularly suitable if you are able to build a community around the software, and encourage third party module

development, to broaden the appeal and reach of your software. This approach is commonly known as “open core”.

This is not without risk, of course, and it might require a shift in business model, to providing support and maintenance services, or to offering custom development and consultancy, but it might keep you in a market which otherwise might not have supported you.

As a desirable sales pitch

One of the biggest costs of any project involving software, alongside licensing costs themselves, is support and maintenance — that, in order to continue to receive patches and updates, and to have someone knowledgeable about the software available to support, you need to continue paying. This form of tie-in is very attractive to many software companies, which seen the ongoing provision of support and maintenance as a continual revenue stream.

To a company looking at purchasing a particular solution, the ongoing costs of support and maintenance may be substantial. Worse, a company may feel tied-in to that vendor: even if their support is poor, or the ongoing costs high, there may be little or no alternative.

You may find it highly marketable to be able to explain to such a company that, whilst you can provide support and maintenance, and, as the author of the code base, are very well placed to do so, since you make the code available under FOSS terms, anyone can look at the code base, and make the modifications that the company might want in the future: they can use you if they wish, but that there is no compulsion to do so.

In this way, whilst you might be introducing competition which you might not have otherwise had, you may open up potential business which you might not otherwise have won: if a company feels that they will be able to support the software if they no longer wish to deal with you, either through in-house support or through engagement of a consultancy of their choice, using your software, and taking your services for as long as they wish, may be desirable.

Why might a business use third party FOSS software?

Unlike a decision to license one's own code base under FOSS terms, the rationale for the use of third party FOSS is straightforward: to enable the business to focus on differentiating elements, rather than recreating components which already exist.

Take, for example, a company which wishes to offer a novel new web service. The company could, if it so wished, build its own infrastructure from the ground upwards — it could write its own operating system, and web server, and supporting infrastructure.

However, it is likely that perfectly good code for these functions exists already, and effort spend on developing these components, which are unlikely to differentiate the company's product from others, takes resource away from spending on areas which really could add value.

By using FOSS code for these non-differentiating elements, not only can resource be focussed on developing differentiating components, changes to the supporting infrastructure can be made easily, and by a wider range of consultants than may be the case in respect of proprietary software: the source code is available for inspection by anyone with sufficient skills.

In a similar vein, in the event of problems, you are not dependent on a third party deciding to work on the problem and issue a patch: you can look to patch the problem yourself, pay someone to do it for you, or see whether someone else in the user community has already done so. In doing this, reliance on a single point of failure is reduced.

Other "opens"

Free and open source software is not the only "open" approach to licensing. Other "opens" include open data, open hardware, as well as the popular licensing scheme, Creative Commons.

For more information on these, see Andrew Katz's chapter, "*Open Everything*", in Shemtov and Walden's "*Free and Open Source Software*".

Creative Commons

Creative Commons is a public foundation which has developed a series of copyright licences which facilitate sharing and co-creation — if the default restrictions of copyright are too onerous for your intended use, rather than drafting a licence from scratch, you could use one of the pre-written Creative Commons licences.

Each licence comes in three forms: a simple, human-readable form; a fuller, lawyer-readable form; and a machine-readable form.

The human-readable form is a summary of the licence, outlining the key terms. The idea is that someone who is not a lawyer can get a good idea of the key terms just by reading the summary. You can see an example of the summary here: https://creativecommons.org/licenses/by/4.0/

The lawyer-readable form is known as the legal code, and is a what a lawyer would expect of a copyright licence: they are clearly written, but quite long. You can see the legal code equivalent of the summary above, here: https://creativecommons.org/licenses/by/4.0/legalcode

Lastly, there is the machine readable form of the licence, expressed in the Creative Commons Rights Expression Language.[81] This is designed to be "a summary of the key freedoms and obligations written into a format that software systems, search engines, and other kinds of technology can understand",[82] to effect easy reuse of Creative Commons works.

All of the main Creative Commons licenses require attribution: this is referenced by the use of the "BY" descriptor. As such, all Creative Commons licence designations begin "CC BY".

There are three other options (or restrictions) which can be added to create the flavour of licence which suits your aim the best: share-alike (SA), non-commercial use only (NC) and no derivatives (ND).

---

[81] https://creativecommons.org/licenses/by/4.0/legalcode

[82] https://creativecommons.org/licenses/

The current version of the Creative Commons licences is version 4; the descriptions which follow come from version 4. Previous versions may contain differences.

SA: share-alike: this introduces a copyleft mechanism into the licence, akin to that described above. Where a licence incorporates the "share-alike" condition, any modifications made to the covered work must be made available under the same Creative Commons terms. This is handled within section 3 of the legal deed:

> If You Share the Licensed Material (including in modified form), You must … indicate the Licensed Material is licensed under this Public License, and include the text of, or the URI or hyperlink to, this Public License.

NC: non-commercial: this prevents the covered work from being used in a commercial context. Commercial use is defined as:

> "not primarily intended for or directed towards commercial advantage or monetary compensation. For purposes of this Public License, the exchange of the Licensed Material for other material subject to Copyright and Similar Rights by digital file-sharing or similar means is NonCommercial provided there is no payment of monetary compensation in connection with the exchange."

ND: no derivatives: this condition prohibits the sharing of modified versions of the work. A recipient is permitted to "produce and reproduce, but not Share, Adapted Material."[83]

This is a variation of the previous definition of "no derivatives"; under version 2 of the licences, the "ND" condition prohibited the creation of derivative works, rather than just their sharing,[84] and, in version 3.0, the permissions granted "include the right to make such modifications as are technically necessary to exercise the

---

[83] See section 2(a)(1)(B): http://creativecommons.org/licenses/by-nd/4.0/legalcode

[84] See section 2 of https://creativecommons.org/licenses/by-nc-nd/2.0/uk/legalcode

rights in other media and formats, but otherwise you have no rights to make Adaptations".[85]

The licences are easily referenced by their short codes:

- The most permissible licence is simply CC BY: it requires attribution, and imposes no further restrictions.

- A licence which permits the creation of derivative works, and permits exploitation on a commercial basis, but requires adaptations to be shared is listed as CC BY SA.

- The most restrictive licence, incorporating all the conditions, is CC BY NC ND; there is no requirement for an "SA" condition here, since the prohibition on the creation of derivatives prevents there being anything to share-alike.

To avoid confusion as to which version of the licence applies, it is good practice to list the version number after the licensing coding. e.g. CC BY NC ND 4.0.

In addition to the options for copyright licensing, Creative Commons offers a mechanism to attempt to place a work in the public domain: CC0. CC0 is the shortest licence in the suite, and is designed to commit a work to the public domain, disclaiming all the author's rights in the work, both copyright and other, wider, rights — this includes the right to attribution.

CC0, as a public domain dedication, is unlikely to operate effectively under English law, since it is not permitted to disclaim a property right: property must be owned.

Under English law, all property must have an owner – the title to a beneficial interest must be vested in someone. There is a body of law in place to deal with the ownership of property by someone who dies intestate, or where a company is liquidated – reversion via *bona vacantia* to the Crown. As such, even if I do not wish to encumber my own intellectual creation with copyright, I am left

---

[85] See section 3: https://creativecommons.org/licenses/by-nc-nd/3.0/legalcode

with no choice but to license it under the broadest possible terms, as I cannot disclaim it.[86]

Other jurisdictions contain similar rules, and this situation is recognised by CC0: where it is not possible for an author to disclaim all of her rights, CC0 attempts to license those rights in the most permissive manner possible.[87]

The public domain

For more on the public domain, see James Boyle's outstanding book "*The Public Domain*",[88] which is available under Creative Commons CC BY NC SA 3.0.

---

[86] See further Neil Brown, "*iPad — harbinger of the public domain?*", https://www.scl.org/site.aspx?i=ed16955

[87] See " What kinds of rights am I surrendering when I use CC0? ", in http://wiki.creativecommons.org/CC0_FAQ

[88] http://www.thepublicdomain.org/

Annex 1: The Open Source Definition

From http://opensource.org/osd

Introduction

Open source doesn't just mean access to the source code. The distribution terms of open-source software must comply with the following criteria:

1. Free Redistribution

The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale.

2. Source Code

The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost preferably, downloading via the Internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed.

3. Derived Works

The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

4. Integrity of The Author's Source Code

The license may restrict source-code from being distributed in modified form only if the license allows the distribution of "patch

files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.

5. No Discrimination Against Persons or Groups

The license must not discriminate against any person or group of persons.

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.

7. Distribution of License

The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

8. License Must Not Be Specific to a Product

The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.

9. License Must Not Restrict Other Software

The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.

10. License Must Be Technology-Neutral

No provision of the license may be predicated on any individual technology or style of interface.