

Open Source Software Licensing: A Short FAQ

Andrew Charlesworth and Anna Home - Centre for IT & Law, University of Bristol
 Researchers, JISC Study to explore the legal and records management issues
 relating to the concept of the Lifelong Learner Record

Q: The JISC Invitation to Tender that I've just received says that any software created under the project should be licensed under an Open Source Licence that promotes its reuse - I have no idea what this means. What is an Open Source Licence?

A: Most commercial software business models involve licensing end-users to use programs, but denying them access to the program source code, as many software firms consider this to be proprietary. End users are thus licensed to use the binary code, but not permitted to decompile programs, or otherwise access the source code, and may not modify or otherwise alter the source code without the explicit permission of the copyright holder in that source code - this can be described as the 'closed source' licensing model.

Microsoft Office is distributed under a 'closed source' End User Licensing Agreement (EULA) which states amongst other things that:

You may:

- (a) install and use a copy of the Software on one personal computer or other device; and
- (b) install an additional copy of the Software on a second, portable device for the exclusive use of the primary user of the first copy of the Software.

[...]

You may not reverse engineer, decompile, or disassemble the Software, except and only to the extent that such activity is expressly permitted by applicable law notwithstanding this limitation.

Source code has not always been regarded as proprietary, and some people think it should be made freely available, claiming that this will promote more rapid and efficient software development. As a result, there is an increasing trend amongst sectors of the software development community to make programs available under licences which explicitly require that the source code of software be made accessible to all users. In this case, the copyright holder (or rightholder) permits end users to be able to freely use, modify, and redistribute that source code - this can be described as the 'open source' licensing model. OpenOffice is distributed under an 'open source' licence (the GNU Lesser General Public Licence) which provides amongst other things that:

You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

[...]

You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License.

Using an open source licence, or agreement, thus means that other people can use, modify and create new works from the software that you have created without having to ask permission to do so, as long as they obey the licence conditions that you set.

Q: Is there just one 'Open Source Software' licence?

A: No, there are a range of types of open source licences, with very important differences - these will be discussed below. It is worth noting that the term 'open source' has entered widespread use, and in consequence its meaning has been somewhat blurred. However, a key definitional source is the Open Source Definition document maintained by the Open Source Initiative (OSI), a non-profit corporation. Software licences which meet the terms of the OSI Open Source Definition can be certified by OSI and display the OSI Certified Open Source Software certification mark. A list of currently certified licences available for adoption is held at: <<http://www.opensource.org/licenses/index.html>>. If you wish to create your own open source licence which meets the requirements of the OSI Open Source Definition, you can apply to have the licence certified by OSI at: <http://www.opensource.org/docs/certification_mark.php>.

The Open Source Definition (Version 1.9) <<http://www.opensource.org/docs/definition.php>>

The indented, italicized sections below appear as annotations to the Open Source Definition (OSD) and are not a part of the 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.

Rationale: By constraining the license to require free redistribution, we eliminate the temptation to throw away many long-term gains in order to make a few short-term sales dollars. If we didn't do this, there would be lots of pressure for cooperators to defect.

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.

Rationale: We require access to un-obfuscated source code because you can't evolve programs without modifying them. Since our purpose is to make evolution easy, we require that modification be made easy.

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.

Rationale: The mere ability to read source isn't enough to support independent peer review and rapid evolutionary selection. For rapid evolution to happen, people need to be able to experiment with and redistribute modifications.

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.

Rationale: Encouraging lots of improvement is a good thing, but users have a right to know who is responsible for the software they are using. Authors and maintainers have reciprocal right to know what they're being asked to support and protect their reputations.

Accordingly, an open-source license must guarantee that source be readily available, but may require that it be distributed as pristine base sources plus patches. In this way, "unofficial" changes can be made available but readily distinguished from the base source.

5. No Discrimination Against Persons or Groups

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

Rationale: In order to get the maximum benefit from the process, the maximum diversity of persons and groups should be equally eligible to contribute to open sources. Therefore we forbid any open-source license from locking anybody out of the process.

Some countries, including the United States, have export restrictions for certain types of software. An OSD-conformant license may warn licensees of applicable restrictions and remind them that they are obliged to obey the law; however, it may not incorporate such restrictions itself.

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.

Rationale: The major intention of this clause is to prohibit license traps that prevent open source from being used commercially. We want commercial users to join our community, not feel excluded from it.

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.

Rationale: This clause is intended to forbid closing up software by indirect means such as requiring a non-disclosure agreement.

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.

Rationale: This clause forecloses yet another class of license traps.

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.

Rationale: Distributors of open-source software have the right to make their own choices about their own software. Yes, the GPL is conformant with this requirement. Software linked with GPLed libraries only inherits the GPL if it forms a single work, not any software with which they are merely distributed.

10. License Must Be Technology-Neutral

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

Rationale: This provision is aimed specifically aimed at licenses which require an explicit gesture of assent in order to establish a contract between licensor and licensee. Provisions mandating so-called "click-wrap" may conflict with important methods of software distribution such as FTP download, CD-ROM anthologies, and web mirroring; such provisions may also hinder code re-use. Conformant licenses must allow for the possibility that (a) redistribution of the software will take place over non-Web channels that do not support click-wrapping of the download, and that (b) the covered code (or re-used portions of covered code) may run in a non-GUI environment that cannot support popup dialogues.

Copyright © 2005 by the Open Source Initiative

It should be noted that it is perfectly possible to create a licence which permits end users to be able to freely use, modify, and redistribute software source code, but which also places restrictions on the use of the code that are contrary to the Open Source Definition. However, such a licence cannot be identified as OSI Certified, or use the OSI text or graphical certification marks. Use of those marks for software not distributed under an OSI approved licence will be an infringement of OSI's rights in those certification marks. It is also likely that other software developers will not recognise a non-OSI conformant licence as being truly 'open source'.

There are also concerns in the open source software developer community about the proliferation of different open source licences. The argument is that the wider the range of licences, the more difficult it is for developers to ensure that they are in conformity with the all the licences that may attach to open source code in their projects. As such, from the perspective of OSI, it is considered desirable, wherever possible, to use an appropriate existing licence rather than to develop a new one.

Q: A colleague says that using open source licensing means that we don't have to worry about copyright issues, is this true?

A: No, while this is quite a common belief, open source licences actually rely on copyright law for their enforceability. Under copyright law, the rightholder in a copyrighted work, including computer program source code (which is treated as a literary work under UK copyright law), is granted certain exclusive rights in that work, which means their permission must be obtained before anyone else can exercise those rights.

The exclusive rights include:

- The right to reproduce the work (subject, in the UK, to the fair dealing defences and library permissions)
- The right to authorise translations of the work.
- The right to authorise arrangements or other types of adaptation to the work.

A rightholder can assign (transfer) their exclusive rights as a bundle, or as individual rights, to another party, in which case the other party takes on the role of rightholder and controls the exclusive rights. Alternatively, instead of assigning some or all of the rights, the rightholder can license them.

Under a licence agreement, the rightholder grants permission to others to exercise certain rights with regard to the work, but retains overall control of it. The lawful use of the work by the other party (the licensee) is conditional on observance of the licence terms imposed by the rightholder (the licensor). In principle, the rightholder, as licensor, can impose such conditions as they wish upon the exercise of their exclusive rights by a licensee, to the extent recognised and enforceable under domestic laws, e.g. licence agreements that contain unduly onerous or unfair terms may not be enforceable, and licence agreements which require unlawful acts of the licensee will not be enforceable.

Open source licences use the licensing ability of the rightholder not to ensure that source code remains hidden from end-users, but to ensure that it is accessible, freely redistributable, capable of modification and use in derived works, and that those modifications and derived works are capable of being distributed under the same terms as the licence of the original software.

Because open source software licensing relies on copyright law, it is vitally important to pay attention to issues such as who owns the rights in a particular piece of code. If you, or your institution, do not own the rights to a piece of newly created source code, you cannot license it under an open source licence, because you do not have the necessary exclusive rights to do so.

Q: This all sounds a bit complex, can you provide some examples?

A: Let us take a JISC project which intends to create some ePortfolio software. In our first two examples, the software source code under consideration is entirely new, having been created in the course of the project with no code incorporated from any other source.

Example 1

Alice, an employee of Brookside University, creates a software module for the ePortfolio project as part of her job. In the absence of any agreement with JISC concerning copyright, UK copyright law says that Alice is the author of the work but, as it is created in the course of her employment, the copyright owner will be Brookside University. As rightholder, Brookside University can exercise those exclusive rights granted by copyright over the software source code. Brookside University thus has the discretion to license the source code to third parties as 'open source' or 'closed source'.

Example 2

Charlie, an independent contractor, creates a software module for the ePortfolio project under a contract with Brookside University. In the absence of an assignment of copyright from Charlie to Brookside University or to JISC, the copyright vests in Charlie as the author of the work. Because Charlie is a contractor, the employee/employer rule that applied to Alice's work does not apply here. If the copyright is owned by Charlie, the University cannot license the software to third parties as 'open source' or 'closed source' because it does not own the necessary exclusive rights under copyright law. Any decision on licensing to third parties will be up to Charlie. This demonstrates why it is important to sort out the issue of intellectual property rights in works created in the course of projects, prior to their creation.

The following four examples consider outcomes where pre-existing source code is incorporated into the project software.

Example 3

Dymphna, an employee of Brookside University, creates a software module for the ePortfolio project as part of her job, and incorporates source code created by Electra, another employee of Brookside University, for a previous University project. As both Dymphna and Electra are University employees, and have created their works in the course of their employment, the copyright owner in those works will be Brookside University. As rightholder, Brookside University can exercise those exclusive rights granted by copyright over the software source code. Brookside University thus has the discretion in both cases to license the source code to third parties as 'open source' or 'closed source'.

Example 4

Fatima, an employee of Brookside University, creates a software module for the ePortfolio project as part of her job, and incorporates source code she has reverse engineered from a 'closed source' program, the rights in which are owned by Glasscode Ltd. Glasscode's licence for the program states that 'You may not reverse engineer, decompile, or disassemble the Software, except and only to the extent that such activity is expressly permitted by applicable law notwithstanding this limitation.' Fatima's action, in reverse engineering the code, is not covered by any applicable UK law. The University may license to third parties as 'open source' or 'closed source' that part of the source code for which Fatima is the author, as it is created in the course of her employment, meaning Brookside University owns the copyright. However, the University may not license the source code derived from Glasscode's program, as it is not the copyright owner of that source code. Additionally, Fatima may face legal action by Glasscode for reverse engineering their program in breach of Glasscode's licence agreement, and the University may face legal action by Glasscode for incorporating Glasscode's proprietary source code in their project software (a derivative work) in breach of Glasscode's exclusive rights.

Example 5

George, an employee of Brookside University, creates a software module for the ePortfolio project as part of his job, and incorporates source code which the University has licensed from HayleyTech Ltd. HayleyTech's licence agreement is not an open source licence, as HayleyTech considers its source code to be proprietary and it retains the right to license the source code to other third parties on the terms and conditions of its choice. The University may license to third parties as 'open source' or 'closed source' that part of the source code for which George is the author, as it is created in the course of his employment, meaning Brookside University owns the copyright. In the absence of an explicit agreement from HayleyTech permitting it to do so, the University may not license to third parties the source code it has licensed from HayleyTech, as it is not the copyright owner of that source code. The University will also have to consider carefully the terms of HayleyTech's licence agreement to ensure that its proposed distribution of the software module's binary code containing HayleyTech's source code to third parties is permissible under that licence.

Example 6

Ian, an employee of Brookside University, creates a software module for the ePortfolio project as part of his job, and incorporates source code he has downloaded from an open source project. The downloaded source code is covered by an open source licence. The terms under which the University can license the source code in the module will depend in part upon what type of open source licence covers the downloaded source code that Ian incorporated into the module.

Q: So there are different types of open source software licence?

A: Yes, while there are various ways that open source licences can be categorised, they can be broadly divided into three main categories - permissive licences, persistent licences, and persistent & inheritable licences (Oksanen, V. & Välimäki, M., 2002).

Permissive Licences

A permissive licence allows third parties (the licensees) to copy, modify and distribute the source code covered by the licence with limited qualifications, such as requiring that those licensees identify that open source software code has been used, and crediting the developers of that open source code. Permissive licences impose no limitations as to the licensing and exploitation of any proprietary software created by licensees, so they can incorporate the code, or modifications of it, in a commercial 'closed source' product subject to standard commercial licensing terms. For example, Microsoft has incorporated source code licensed under the BSD permissive licence into versions of Windows, including XP, and this places no obligations on Microsoft other than to acknowledge BSD's copyright notice, its list of licence conditions and its disclaimer in the documentation and/or other materials provided with the distribution. The effect of distributing source code under a permissive licence is not the same as placing the software in the public domain, as public domain software is, by its very nature, outside the scope of copyright and licensing. However, the use of permissive licences significantly reduces the control of rightholders over the further use of their source code.

See further <<http://www.oss-watch.ac.uk/resources/modbsd.xml>>

Example of a permissive licence - Modified BSD licence

Copyright (c) 1998, Regents of the University of California
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the University of California, Berkeley nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS AND CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Persistent Licences

A persistent licence allows third parties (the licensees) to copy and distribute the licensed source code, if it is bundled as a whole into a new work, i.e. as a combined or linked work, without requiring that the new work be licensed in its entirety under an open source licence. In such circumstances, licensees are permitted to combine the licensed code with a separate program and distribute the combination under separate licences - the 'open source' code being distributed under the 'open source' licence - the persistence principle. An example of this is where software libraries are compiled into an application program. However, where the licensed source code is modified, resulting in a derivative work, this derivative work must be licensed either under the same, or a stricter, open source licence (the stricter licence usually being the GNU General Public Licence (GPL) see below). The important element of persistent licences is thus that as long as the licensed source code is compiled, or links, with the other work without change, any original source code created by the licensee can be licensed under any terms recognised and enforceable under domestic laws, including a commercial proprietary licence.

The classic persistent licence is the GNU Lesser General Public Licence (LGPL) <<http://www.gnu.org/copyleft/lesser.html>>. A key issue to note is that if a licensee uses LGPL-licensed source code to create a derivative work, rather than a combined or linked work, then the effect of the LGPL is essentially that of a persistent & inheritable licence (see below)

See further <<http://www.oss-watch.ac.uk/resources/lgpl.xml>>

Persistent & Inheritable Licences

A persistent & inheritable licence allows third parties (the licensees) to copy, distribute, and modify the licensed source code, but all bundled and derivative works created by the licensee using the licensed source code must be licensed under the same terms as those which the original licensor applied to their licensees. In effect, a persistent & inheritable licence can create a chain of licensees, all legally bound to ensure that the bundled and derivative works that they have created using source code originally licensed under the first persistent & inheritable licence are covered by an identical licence - thus the 'open source' code retains its 'open source' licence - the persistence principle, and the terms of that licence apply to new works incorporating that source code - the inheritable principle.

The most widely used persistent & inheritable licence is the GNU General Public Licence <<http://www.gnu.org/licenses/gpl.html>>. It appears that the intended effect of the GPL is that a licensee cannot create a derivative or combined work using GPL-licensed source code, without in turn licensing that work under the GPL. However, the wording of the GPL licence leaves room for debate about the extent to which the linking of GPL-licensed source code with non-GPL-licensed source code requires the latter to inherit the GPL licence terms - there appears to be uncertainty, in particular, with dynamic linking (Dixon, 2004)

See further <<http://www.oss-watch.ac.uk/resources/gpl.xml>>

The Mozilla Public Licence

A fourth kind of licence - the commercial open source licence - has also seen an increase in use. A key example of this development is the Mozilla Public Licence (MPL) which is the model for most of the commercial open source licences which have followed it. It is a considerably longer licence agreement than most of the earlier open source licences, including the BSD, LGPL and GPL licences. In legal terms, it is also drafted very much more precisely

than those licences. In effect, the MPL appears to be designed to incorporate key ideas from both the BSD and GPL without being as permissive as the former, or as restrictive as the latter.

Put simply, the MPL is concerned, not with potentially derivative works as a whole, but with the components of derivative works - files. This narrowing of focus permits the development of a licensing structure which allows the creation of works out of a combination of open source components - files containing the original source code licensed under the MPL, or the original source code plus modifications to the code made by licensees - with potentially proprietary components - files containing wholly original code not covered by the MPL. Under the MPL, such a hybrid work is not automatically deemed to be open source, as it would be under the GPL; instead, the files that contain original MPL-licensed source code, and/or modifications of that source code by licensees, must be licensed under the MPL and files that are the original work of a licensee can be licensed under whatever licence arrangements the licensee chooses.

As a result, it is possible to combine MPL-licensed source code in one set of files with closed source code in another set of files, and as an end result create a larger work that can be dealt with as a proprietary product, despite the fact that it contains MPL-licensed source code. Distribution of the larger work as binary code will, of course, still require attention to the MPL conditions, as files within that work which fall under the MPL must be made available as detailed in the licence, and other documentation and notices must be included.

See further <<http://www.oss-watch.ac.uk/resources/mpl.xml>>

Q: So, if a project programmer has used open source-licensed code in a software module for our project, this may affect the way in which we can license the resulting software?

A: Yes, returning to example 6, we can see how this scenario might play out in licensing terms.

Example 6a

Ian, an employee of Brookside University, creates a software module for the ePortfolio project as part of his job, and incorporates some source code he has downloaded from an open source project. The downloaded source code is covered by the modified BSD open source licence. The University may license the resulting software module under an open source or closed source licence, as long as they observe the requirements as to display of copyright notices, licence conditions and disclaimer in source code, or binary code, distributions, as they own the work created by Ian in the course of his employment, and the BSD licence does not impose any additional conditions.

Example 6b

Ian, an employee of Brookside University, creates a software module for the ePortfolio project as part of his job, and incorporates some source code he has downloaded from an open source project. The downloaded source code is covered by the LGPL open source licence. Ian combines the LGPL-licensed source code in unmodified form with source code that he has created, to form an executable. As Ian has not created a derivative work using the LGPL-licensed source code, the University may distribute the software module under two licences - the LGPL licence for the licensed source code, and an open source or closed source licence for the work created by Ian in the course of his employment.

Example 6c

Ian, an employee of Brookside University, creates a software module for the ePortfolio project as part of his job, and incorporates some source code he has downloaded from an open source project. The downloaded source code is covered by the LGPL open source licence. Ian modifies the LGPL-licensed source code, thereby creating a derivative work, and combines it with source code that he has created. By modifying the LGPL-licensed source code Ian triggers the requirement that the entire resulting work (e.g. the software module) be licensed to third parties under the GPL.

Example 6d

Ian, an employee of Brookside University, creates a software module for the ePortfolio project as part of his job, and incorporates some source code he has downloaded from an open source project. The downloaded source code is covered by the GPL open source licence. Ian modifies the GPL-licensed source code and combines it with code that he has created. In doing so, Ian has created a derivative work and, under the terms of the GPL licence, the resulting software module must also be licensed to third parties under the GPL.

Example 6e

Ian, an employee of Brookside University, creates a software module for the ePortfolio project as part of his job, and incorporates some source code he has downloaded from an open source project. The downloaded source code is covered by the GPL open source licence. Ian uses the GPL-licensed source code in unmodified form and combines it with code that he has created. Although Ian has not created a derivative work, the act of combining the GPL-licensed source code with his code means that, under the terms of the GPL licence, the resulting software module must also be licensed to third parties under the GPL.

Q: So if the University owns the copyright in the source code, does this mean that the choice is between closed source and open source software licensing?

A: Not necessarily, there is a third option that can be considered. If the University is the rightholder in the source code, it is in a position to determine both what the licence terms will be, and also to whom those terms will apply. For example, software companies often license software to educational institutions at a lower price, or with fewer restrictions than they license the same software to commercial firms. This is because the law permits them to distinguish between their treatment of different categories of licensee. Equally, the University might consider dual licensing the software in which they own the rights to the source code. This means that the University could offer the source code under a GPL licence to developers who were happy to share any modifications of the code or derivative works under the same licence conditions, and a separate non-open source licence for a fee to commercial bodies wishing to include the source code in proprietary closed source software.

A number of software companies use precisely this model, for example the open-source database companies MySQL and Sleepycat, and programming component maker Trolltech, make their source code available under an open-source licence for use in open-source software, and under a commercial licence for inclusion in proprietary software. It is important to remember, however, that the dual licensing strategy will only work where the University owns the copyright to all the source code in the software package.

Example 7

Jasinder, an employee of Brookside University, creates a software module for the ePortfolio project as part of his job. In the absence of any agreement with JISC concerning copyright, UK copyright law says that Jasinder is the author of the work but, as it is created in the course of his employment, the copyright owner will be Brookside University. As rightholder, Brookside University can exercise those exclusive rights granted by copyright over the software source code. The University thus has the discretion to license the source code to third parties as 'open source' or 'closed source'. The University decides to make the software module available under the GPL so that other educational users can both use and help develop the module.

The University is then approached by Kimberley Software Ltd which wishes to use the software module in a commercial ePortfolio system. The University licenses the software module source code to Kimberley Software for £5000 plus a percentage of gross profits, under a licence which permits Kimberley Software to use, modify and distribute the source code in its commercial product without the open source constraints.

The University cannot license the improved code generated by other contributors under the open source licence to Kimberley Software, as the University does not own the copyright to those improvements. The other contributors under the open source licence cannot license their improvements to the code to Kimberley Software in this way as those improvements are derived works of the original code licensed to them by the University under the GPL. Thus, any licence granted by the other contributors would also be a GPL licence, and Kimberley Software are unlikely to want to incorporate GPL-licensed code into their product as this would probably result in the GPL licence terms applying to Kimberley Software's ePortfolio product.

Q: I am a project manager, and I'm considering whether to make the project software open source, how should I choose which of the different licences to use?

A: The choice of software licence is an important decision, and should ideally be made before the software is developed, in consultation with your funders, your institution, and where appropriate with your institution's lawyers. A key determinant of any licensing decision is going to be who owns the copyright in the source code which is to be licensed, and what, if any, prior agreements (e.g. with funders) have been made as to licensing and distribution.

Example 8

Ledley, an employee of Brookside University, creates a software module for a JISC-funded ePortfolio project as part of his job. UK copyright law says that Ledley is the author of the work but, as it is created in the course of his employment, the copyright owner will be Brookside University. The funding agreement that the University has signed with JISC states that all software outputs of the project will be owned by the institution but must be licensed to other UK FE and HE institutions in a way that that permits them to copy, modify and distribute the software within the FE and HE sectors.

Ledley wants to make the software source code available under the BSD permissive licence. However, his project manager, Miranda, knows that this will mean that commercial firms can incorporate the source code into their proprietary products as long as they observe the requirements concerning display of the University's copyright notices, licence conditions and disclaimer in source code, or binary code, distributions. Miranda would rather license the source code under the GPL as she feels this will dissuade commercial firms from 'free-riding' on the University's work, and will encourage other developers across the HE and FE sectors to get involved with the development.

Nigel, the University Research and Development Officer, would prefer the software to be developed by a University spin-out company and sold only as a closed source proprietary binary distribution, however while the University owns the copyright in the work, the contractual terms of the JISC funding agreement prevents a wholly proprietary approach. Nigel would thus like to dual license the source code making the software available to UK FE and HE institutions under the GPL, whilst granting a non-open source licence to a University spin-out company to develop an enhanced closed source proprietary binary distribution.

Ollie, a new member of the University Research and Development Office, suggests that rather than using the GPL, the University should create its own licence, the Brookside Licence, based on the MPL. This would permit the University to meet its funding agreement obligation to JISC to distribute the source code to other FE and HE institutions in the UK, allow the University spin-out company access to any useful source code developments created by other developers under the Brookside licence, and leave room for the spin-out company to develop proprietary source code files to enhance a proprietary binary software distribution. However, Peter, another member of the R & D Office with experience of open source development, notes that creating a new open source licence can be extremely difficult. It will not be regarded by many developers as truly 'open source' unless it is certified by the OSI and the granting of such certification is rare, takes considerable effort, and often ends in frustration. He advises using a licence that has already been certified, if at all possible.

The decision as to how to license a particular piece of software source code may thus be affected by many variables, including personal preferences amongst your project team or other institutional officials, your institutional research and development strategy, your funder or sponsor preferences or requirements, assessments of the likelihood that the software will or will not attract a viable open source development community, and the amount of time that you or your project team have to manage an long-term open source development project. It is also worth considering, in the event that your institution is in a position to consider commercial exploitation of software source code, the effect that licensing the source code under an open source licence may have on third party willingness to invest in further development. For small amounts of source code, this decision may not be particularly difficult; however, for a substantial piece of software development it may well be cost-effective to seek professional legal advice. This approach is particularly advisable in circumstances where your institution is considering creating a new open source licence, as the OSI are actively attempting to dissuade further licence variations, and thus it is likely to be very difficult to devise a new form that will be certified by the OSI.

Q: Does this mean that if my project team are developing software as part of the project, I not only need to be planning ahead in terms of possible licensing strategies on the basis of all these variables, but I also need to make sure we know the provenance of any source code they incorporate into the project from third parties?

A: Yes, as we can see from the examples throughout this FAQ, it is vital to know the provenance of all source code used in a piece of software. If your institution does not own the rights to the source code, it can make the task of determining the appropriate licence for the source code both more difficult, and potentially increase the legal risks. Developers on your team should be clear about what your project's licensing goals are, and how their activities might affect those goals. For example, incorporation of GPL-licensed software source code into a project which your institution or a funder wishes to keep proprietary, may result in: inadvertent breach of the GPL and attendant bad publicity; the need to analyse the software in detail in order to remove GPL-licensed source code and replace it with new code, with all the costs that may entail; and, in the event that you are unable to vouch for the provenance of your code, the

loss of venture or incubator financing, or commercial partnership opportunities. Ideally, project managers should, as a matter of routine, have an IPR register in which the use of any third party source code in software created by the project is noted, along with the details of any licence agreement that attaches to it, so that it is possible to identify potential problems or licence clashes at an early stage.

Q: Our institution has insufficient development expertise in-house, but a good working relationship with independent software developers. What steps do I need to take when out-sourcing open source software development?

A: Any form of outsourcing of software development requires an institution to think carefully, well in advance of contacting an external developer, about the legal implications of doing so, including, but not limited to, the ownership of the intellectual property that may be generated as a result of the outsourcing process. Example 2 above provided a very basic scenario of an outsourcing arrangement. However, a contract term that simply requires a contracted developer to assign the copyright in the software code to the institution may be insufficient to meet the institution's needs. In order for effective use to be made of the code, the institution may also require access to, or ownership of, the developer's design notes, as well as any documentation the developer has created to explain the operation of the code. Additionally, the institution is likely to want protection, in the form of legal indemnities, against intellectual property breaches committed, deliberately or inadvertently, by the developer, such as incorporating unlicensed proprietary code into the software, or incorporating software that is licensed either under proprietary or open source licences that are incompatible with the institution's preferred open source licensing strategy or patent policy.

Matters are complicated further if the institution wants to leverage the contributions of voluntary software developers in an open source project. Here, the institution could ask for a complete assignment of rights from the developers in any new code created for the project in order to ensure control of the necessary rights, but developers might be hesitant to sign away all their intellectual property rights. An alternative might be to request a joint assignment of copyright, so that both the contributor and the institution have ownership rights in the software code. An example of this approach can be seen with OpenOffice, where contributors assign joint copyright to Sun Microsystems, Inc. for their contribution (see the licence agreement at <http://www.openoffice.org/licenses/jca.pdf>). In principle, this gives Sun sufficient rights to relicence the code as they wish, but also permits the contributor to do whatever else they might like to do with the code that they created. Not all code submitted is accepted for inclusion in the project, and in such circumstances the full copyright ownership simply reverts to the contributor. Engaging in such an arrangement will require an institution to maintain an IPR register that contains the assignments, and clearly identifies the code that each assignment attaches to. The licensing process and location of responsibility for maintaining the IPR register should, again, be thought out well in advance of beginning the project.

It is worth noting that while the OpenOffice licence contains clauses that state that the contributor "owns, and has sufficient rights to contribute, all source code and related material intended to be compiled or integrated with the source code for the OpenOffice.org open source product" and "is legally entitled to grant the above assignment and agrees not to provide any Contribution that violates any law or breaches any contract," there is no indemnity clause - unsurprisingly, as developers are unlikely to wish to risk any legal liability for software code which they have effectively donated to an open source project. This lack of indemnity from liability available from the developers means that the institution will have to consider, and if necessary make provision for (via insurance), its potential liability in the event of a claim that the resulting software breaches a third party's intellectual property rights.

Bibliography

- Bonaccorsi, A. & Rossi, C. (2003) *Licensing schemes in the production and distribution of Open Source software. An empirical investigation* <<http://opensource.mit.edu/papers/bnaccorsirossilicense.pdf>>
- Dixon, R. (2003) *Open Source Software Law*, Artech House Books, ISBN: 1580537197.
- Oksanen, V. & Välimäki, M. (2002) *Evaluation of Open Source licensing models for a company developing mass market software.* <http://www.hiit.fi/de/valimaki_oksanen_lawtech_2002.pdf>
- Rosen, L. (2004) *Open Source Licensing: Software Freedom and Intellectual Property Law*, Prentice Hall, ISBN: 0131487876.
- Rossi, M.A. (2004) *Decoding the "Free/Open Source (F/OSS) Software Puzzle" a survey of theoretical and empirical contributions.* <<http://opensource.mit.edu/papers/rossi.pdf>>
- St. Laurent, A. M. (2004) *Understanding Open Source and Free Software Licensing*, O'Reilly, ISBN: 0596005814. <<http://www.oreilly.com/catalog/osfreesoft/book/>>
- Välimäki M. (2003) *Dual Licensing in Open Source Software Industry* <<http://opensource.mit.edu/papers/valimaki.pdf>>

Key websites

- OSS Watch: Open Source Software Advisory Service <<http://www.oss-watch.ac.uk/>>
- Open Source Initiative (OSI) <<http://www.opensource.org/>>
- Open Source Resource Community <<http://opensource.mit.edu/>>
- The Free Software Foundation <<http://www.fsf.org/>>

This document was written in consultation with OSS Watch, the open source software advisory service funded by the Joint Information Systems Committee, and particular thanks are due to Randy Metcalfe, Manager of OSS Watch, for his comments and suggestions - not least for the observation that in UK English, "licence = noun and license = verb".

This document is a work in progress. If you would like to see this FAQ cover other areas of law, address existing areas in more detail, or answer specific questions, or indeed if you would like to contribute to it, please contact the researchers at:

Email: <A.J.Charlesworth@bristol.ac.uk> or <Anna.Home@bristol.ac.uk>

Address: The Law School, University of Bristol, Wills Memorial Building, Queens Road, Bristol BS8 1RJ

New discussion list: EPORT-LEGAL@jiscmail.ac.uk
See <<http://www.jiscmail.ac.uk/lists/EPORT-LEGAL.html>>