Updated Report on license issues, Liability and SW Procurement Policies in Relationship with FOSS

Period: Jan–Jun 2012
Project Partner: K.E.D.E.

Project acronym: OSEPA

Project name: Open Source software usage by European Public Administrations

Project code: INTERREG IVC, 0918R2

Document Information:

Document title: Report on license issues, Liability and SW Procurement Policies in Relationship with FOSS

Date of Delivery: 29.06.2012

Component: CP3

Component Title: Exchange of experience

Component Leader: USFD

Distribution: Restricted to the partners of the Consortium

Nature: Report

History Chart

Date	Changes	Cause of change	Implemented by
30.06.2010	Initial Document	N/A	KEDE
29.06.2012	Updated Document	Evaluate Feedback	KEDE

Authorisation

No.	Action	Partner	Date
1	Prepared	KEDE	29.06.2012
2	Approved		
3	Released		

Disclaimer

The information in this document is subject to change without notice.

Table of Contents

1. license issues

1.1 Pluralism and the importance for Public Administrations

Software licensing is a critical aspect for the public sector in terms of providing the context in which software may be used, distributed or modified. This is particularly important for public administrations that either plan to implement projects requiring re-use or modification of software products and components or even release their own custom-built software solution. Lack of awareness on different license types and licensing issues often leads public administrations to single vendor lock-ins or to a discrimination against open source solutions.

A software licence adoption policy could be based on a number of criteria some of which may be particularly weighted for governments and public administrations. For instance the need for unlimited access to source code or unlimited usage of the software. The right to reproduce and distribute an unlimited amount of copies, the right to modify the software and the right to reproduce and distribute an unlimited amount of copies of the modified software version under the same license restrictions.

The seek of the “right” license reveals the amount of Free Open-Source Software (FOSS from now on) licenses.

They are divided in two many categories. Those approved by the Free Software Foundation (FSF) and those approved by the Open-Source Initiative (OSI). These lists of the approved licenses are periodically updated.

1.1.1 Free as in Freedom

The Free Software Foundation has authored the Free Software Definition and besides of a Free Software repository it also maintains a list of approved licenses.

These licenses are chosen in the sense of preserving the following four freedoms:

- *Use for any purpose.* Practically this means there is no End User license Agreement (EULA) or Terms of Use (TOS). Everyone is free to use the software however he likes.
- *Study, by examining the source code.* The software isn't limited to just a binary. It's accompanied with its source code.
- *Modify and improve.* The availability of the source code is required.
- *Distribute, with or without modifications.* Anyone can distribute his own copy of a free software, even if he has modified it. Note that you should respect the copyright (the moral right) in any case.

The most famous and most frequently used license of this kind is General Public license (GPL). It has reach its third release and its main goal is to protect the above freedoms in software. Beyond these, the most significant characteristic of GPL is that it's a viral license. Every software based on modifications of a GPL one or depending (essential linking for function properly) on it, has to be licensed under GPL as well. All viral licenses are considered as “copyleft”.

Another example is AGPL. It's a license very close to GPL, but designed to close a perceived application service provider "loophole", in the ordinary GPL, where by using but not distributing the software, the copyleft provisions are not triggered. Thus is mostly used for web applications.

1.1.2 The OSI licenses

OSI stands for Open-Source Initiative. It's an organization for promoting Open-Source. It maintains a list of approved licenses that fit the Open-Source Definition. These licenses have some restrictions (compared to Free Software licenses), as they don't have to preserve all four freedoms mentions before. Ussually these licenses are not considered copyleft, due to the lack of the viral “behaviour”.

OSI licenses are common used for escaping the viral clause of GPL, although not all of them allow it, and thus helping the development of proprietary applications based on Open-Source platforms.

Some of the most famous OSI licenses used on Open-Source projects are BSD, MIT, APL. BSD license, used for the BSD operating system, is most known for not having the GPL viral clause,

allowing to anyone who uses a BSD-licensed project to redistribute it with a proprietary license and without the obligation for distributing the source code. The most famous example is the network stack on some Microsoft's operating systems that was based on BSD-licensed FreeBSD, or even Apple's OSX kernel that is based on FreeBSD as well. On both of these examples neither of Microsoft and Apple have released the source code of the modifications.

MIT license and Apache Public License (APL) both allow linking with proprietary software, in contrary with GPL. The most significant difference between these two license is that MIT is compatible with GPL, but APL is not.

Another famous license Mozilla Public license (MPL), that is used on Mozilla Firefox that allows combination with propriety software (in contrary to GPL), although any modifications on the MPL source code should also be distributed as MPL (weak copyleft).

1.1.3 EUPL

EUPL stands for European Union Public license, and it's approved by FSF. It's an effort to bring GPL close to the European legal system. It's widely used and adopted for software projects that aim to be used in the Public Administration. Currently more than 70% of the software that is hosted on Joinup (the EU Commission Open-Source repository) are licensed under EUPL.

1.2 Legal misunderstandings

The most common misunderstanding about all these examples, especially with GPL, is that they don't allow any kind of co-existance with proprietary software. Proprietary software can co-exist with GPL-licensed software with no legal issues. What is prohibited is for a proprietary application linking on a GPL-licensced one. In plain words a proprietary application can't use GPL-licensed software in order to use properly, but they can co-exist in the same platform or operating system.

As mentioned before the three most commonly used Open-Source licenses, and most of the OSI licenses, are allowing linking with proprietary software.

1.3 Dual Licensing

Dual licensing is applied in cases where the same software code is released and distributed under both a free software (e.g. GPL) and a proprietary license. This model offers users an exception from the “copyleft” requirement to release derivatives under the same license type by providing a second, proprietary license option. In this way, licensees (e.g. companies, developers) are able to decide whether they want to adopt an open source or more restrictive approach on controlling software use, distribution and sub-licensing.

The benefit of dual licensing is that it can work both ways. Open source licensing allows external contributions under the same license type while proprietary licensing helps to fund and commercially promote a software product. In dual licensing models the core product is controlled by a single vendor or original developer not allowing for competing or substitution products (forks) to be introduced by developer communities.

It has been shown, however, that this model may also lead to limited external contributions by developers, due to the “same licensing regime” requirement. A typical example of a dual licensing business model is that of MySQL. MySQL provides the option of choosing between GPL and a commercial license. Within this scheme, those producing and distributing FOSS under a “copyleft” license can use the GPL licensed MySQL code. On the other hand, developers or companies who

wish to use the MySQL code but are not willing to release the source code of their own software products may acquire the commercial license.

1.4 licenses Summary

Below is a summary of commonly used licenses. Note that:

- When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library.

- The “Proprietary Software linking” refers to linking or closed sourced applications/libraries with applications/libraries licensed under one of the following Open Source licenses.

- By “the Work” in the “Distribution of 'the Work'” I mean a combination of a software with the library or application licensed under one of the following licenses

- “Redistributing of the code with changes” refers to the act of redistributing a modified app/library based on the app/library licensed under the given license

license	Proprietary Software Linking	Distribution of the work	Redistribution of code changes	Compatible with GPL
GPL	No	Not allowed with software whose license is not GNU GPL compatible	Only if the derivative is GNU GPL	Yes
LGPL	Yes	Allowed with some restrictions: You have to provide source code of the distributed LGPL library with (if any) modifications, changes to the LGPL library should be allowed to third parties and if BC your app/lib should still work with the modified LGPL lib/app	Only if the derivative is GNU LGPL or GNU GPL	Yes
Apache	Yes	Yes	Allowed (as long as the name “Apache” isn't used in the name of the derivative work)	No
BSD	Yes	Yes	Yes	Yes
Mozilla	Yes	Yes	Only under MPL	No
MIT	Yes	Yes	Yes	Yes
Apple	Yes	Yes	Only under Apple Public license	No
Python	Yes	Yes	Allowed, assuming the package includes a list of changes to the original Python and copyright notices on all files	Yes
W3C	Yes	Yes	Yes	Yes

1.5 Shared Licensing Strategy

Public administrations with shared objectives and similar organizational needs should jointly develop “one to serve all” licensing policies for software. In this way they could strongly put forward common wants and needs on software and develop a shared knowledge basis on licensing issues as a firm, common ground for selecting best value for money solutions.

Adopting shared licensing strategies based on common needs and mutual understanding would help public administrations to:

- better serve their software requirements based on their operational tasks and organizational needs.

- maximize the re-usability and transferability of acquired software components and applied solutions

- reduce the cost of software license purchasing and updating.

- promote a uniform software licensing regime across public sector networks or associations (e.g. regional or national associations of municipalities) based on a commonly accepted license type such as the European Union Public License.

2. Liability

In many cases there is a need of legal responsibility on the unlikely event of software failure. For instance if the failure caused data loss. This is what usually called liability.

2.1 licenses disclaimer

It's common ground to Open-Source licenses to renounce legal obligations on software failure. Here is the disclaimer from GPL that describes limitation of liability.

> **16. Limitation of Liability.**
> IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Similar disclaimers exist on every Open-Source license. The reason for this is that an Open-Source project is potentially developed by the whole world, or a little more realistic from a large community. So the above disclaimer is a way to protect the community from legal obligations. Otherwise a user could blame every contributor of a project on a failure incident (eg. data loss). This is short of a legal protection for the community.

2.2 Paradigm shift

As the business model differs in Open-Source, from Software as a product to Software as a Service (SaaS), so does the Software Liability. Any legal obligation are “transferred” from the copyright holder to the service provider, the vendor with the support contract. This new model is something that must be taken in advance when designing Software Procurement Policies.

2.3 Side issues

It's critical to empasize that in Europe there are no software patents (as opposed to USA), so all legal issues derive from common copyright laws.

3. Software procurement policies

3.1 Open standards

Maybe the most critical issue on software selection is the support of Open Standards.

ITU (International Telecommunication Union) defines Open Standards as standards that are made available to the general public and are developed (or approved) and maintained via a collaborative and consensus driven process. "Open Standards" facilitate interoperability and data exchange among different products or services and are intended for widespread adoption.

Other elements of "Open Standards" include, but are not limited to:

- Collaborative process – voluntary and market driven development (or approval) following a transparent consensus driven process that is reasonably open to all interested parties.
- Reasonably balanced – ensures that the process is not dominated by any one interest group.
- Due process - includes consideration of and response to comments by interested parties.
- Intellectual property rights (IPRs) – IPRs essential to implement the standard to be licensed to all applicants on a worldwide, non-discriminatory basis, either (1) for free and under other reasonable terms and conditions or (2) on reasonable terms and conditions (which may include monetary compensation). Negotiations are left to the parties concerned and are performed outside the SDO.
- Quality and level of detail – sufficient to permit the development of a variety of competing implementations of interoperable products or services. Standardized interfaces are not hidden, or controlled other than by the SDO promulgating the standard.
- Publicly available – easily available for implementation and use, at a reasonable price. Publication of the text of a standard by others is permitted only with the prior approval of the SDO.
- On-going support – maintained and supported over a long period of time.

W3C (World Web Consortium), known for defining web standards such as XML, HTML, CSS, claims these prerequisites for Open Standards:

- transparency (due process is public, and all technical discussions, meeting minutes, are archived and referencable in decision making)

- relevance (new standardization is started upon due analysis of the market needs, including requirements phase, e.g. accessibility, multi-linguism)
- openness (anybody can participate, and everybody does: industry, individual, public, government bodies, academia, on a worldwide scale)
- impartiality and consensus (guaranteed fairness by the process and the neutral hosting of the W3C organization, with equal weight for each participant)
- availability (free access to the standard text, both during development and at final stage, translations, and clear IPR rules for implementation, allowing open source development in the case of Internet/Web technologies)
- maintenance (ongoing process for testing, errata, revision, permanent access)

Public Administrations using Open Standards have the ability to avoid Vendor Lock-in.

3.2 Vendor Lock-in

This term is used when a customer is dependent on specific vendor for software and services, unable to use another vendor without substantial switching costs.

The most usual case of Vendor Lock-in is the usage of proprietary and maybe patented standards. For instance the usage of a proprietary file format for storing documents can lead to dependency from a specific (proprietary) Office Suite.

3.3 Exit migration cost

The migration cost is something that always is taken in consideration when it comes to migrate on a different platform or vendor. Many times it's the key factor for not migrating to Open-Source solutions and extending the vendor lock-in dead end.

A more radical approach to migration cost is to evaluate a software solution by the cost needed in order to migrate out of it (exit cost). On long-term period this has more financial benefits.

3.4 Vendor discrimination / Tender preference

Equal treatment of all vendors is essential for Procurement Policies. No vendor should be hard-coded in Procurement Document.

Despite the above every organization can determine its own needs for the IT infrastructure and choose the best form in which an IT solution should be structured, and this includes how it should be paid and accounted for. Some business models and service models are a naturally better fit for given requirements that are determined and defined by a public agency prior to procurement.

All these choices involve discrimination between business models, and a preference for some business models over others. Businesses that use a business model that cannot meet the needs of the public agency will naturally lose out. This is not against equal treatment and non- discrimination mention before.

A good example is the “Netherlands Open Connection” policy which expresses an explicit "preference for open-source software in the case of equal suitability". It recognizes that public procurement must not discriminate between individual vendor , but a preference towards a specific business model is generally accepted and wide-spread in several areas - such as when a preference is expressed for leasing instead of buying capital equipment in a call for tender.