EUROPEAN PARLIAMENT

DIRECTORATE-GENERAL FOR INTERNAL POLICIES

POLICY DEPARTMENT C
CITIZENS' RIGHTS AND CONSTITUTIONAL AFFAIRS

Constitutional Affairs

Justice, Freedom and Security

Gender Equality

Legal and Parliamentary Affairs

Petitions

Legal aspects of free and open source software WORKSHOP Tuesday, 9 July 2013

COMPILATION OF BRIEFING NOTES

EN

2013

EUROPEAN PARLIAMENT

DIRECTORATE GENERAL FOR INTERNAL POLICIES

POLICY DEPARTMENT C: CITIZENS' RIGHTS AND CONSTITUTIONAL AFFAIRS

LEGAL AFFAIRS

Legal aspects of free and open source software

COMPILATION OF BRIEFING NOTES

WORKSHOP
Tuesday, 9 July 2013
JAN 4 Q 1

PE 474.400 EN

This document was requested by the European Parliament's Committee on Legal Affairs.

RESPONSIBLE ADMINISTRATORS

Danai PAPADOPOULOU
Policy Department C: Citizens' Rights and Constitutional Affairs
European Parliament
B-1047 Brussels
E-mail: danai.papadopoulou@europarl.europa.eu

Rosa RAFFAELLI
Policy Department C: Citizens' Rights and Constitutional Affairs
European Parliament
B-1047 Brussels
E-mail: rosa.raffaelli@europarl.europa.eu

EDITORIAL ASSISTANCE

Marcia MAGUIRE
Policy Department C: Citizens' Rights and Constitutional Affairs

LINGUISTIC VERSIONS

Original: EN

ABOUT THE EDITOR

To contact the Policy Department or to subscribe to its monthly newsletter please write to:
poldep-citizens@europarl.europa.eu

European Parliament, manuscript completed in July 2013.
© European Union, Brussels, 2013.

This document is available on the Internet at:
http://www.europarl.europa.eu/studies

DISCLAIMER

CONTENTS

LIST OF ABBREVIATIONS

AGPL	Affero General Public License
APL	Apache Public License
BSD	Berkeley Software Distribution
EC	European Commission
EIF	European Interoperability framework
EPL	Eclipse Public Licence
EU	European Union
EUPL	European Union Public Licence
FDL	Free Documentation License
FOSS	Free and/or Open Source Software
FLA	Fiduciary Licensing Agreement
FLOSS	Free/Libre Open Source Software
FRAND	Fair, Reasonable and Non Discriminatory (licensing terms)
FSF	Free Software Foundation
GCC	GNU C Compiler
GNU	GNU's Not Unix
GPL	GNU General Public License
IMIO	Intercommunale de Mutualisation Informatique et Organisationnelle
IP	Intellectual Property
ISA	Interoperable Solutions for Administrations (a EC programme, complementing the previous IDA and IDABC)
ISV	Independent Software Vendor
IT (ICT)	Information (& Communication) Technology

KDE	K Desktop Environment
LGPL	Lesser General Public License
MPL	Mozilla Public Licence
NIF	National Interoperability Framework
NOIV	Nederland Open In Verbinding
OSI	Open Source Initiative
OSS	Open Source Software (see: FOSS)
SaaS	Software as a Service (i.e. applications used in the cloud)
PAs	Public Administrations
SDO	Standard Definition Organisation
SFLC	Software Freedom Law Center
W3C	World Wide Web Consortium

An introduction to the most used FOSS license: the GNU GPL license

Dr. Eben Moglen, JD and Ian Sullivan,
Columbia Law School

ABSTRACT

The public drafting and discussion of GPLv3 in 2006-07 was a landmark in non-governmental transnational law-making. Free and open source software production communities are held together by copyright licensing, as are free cultural production communities like Wikipedia. Their efforts to improve those licenses—to increase their utility in multiple legal systems, to take account of technical and economic changes in the field, and to increase their efficiency of operation and enforcement—are among the most important examples of genuinely democratic, participatory law-making that we have experienced so far in the 21st century.

CONTENT

EXECUTIVE SUMMARY

The public drafting and discussion of GPLv3 in 2006-07 was a landmark in non-governmental transnational lawmaking. Free and open source software production communities are held together by copyright licensing, as are free cultural production communities like Wikipedia. Their efforts to improve those licenses—to increase their utility in multiple legal systems, to take account of technical and economic changes in the field, and to increase their efficiency of operation and enforcement—are among the most important examples of genuinely democratic, participatory law-making that we have experienced so far in the 21st century. In the interest of improving both the European Parliament's access to the details of this particular process, and to assist it in self-scrutiny, with respect to its extraordinary consistency in missing its opportunities in this area, Software Freedom Law Center (SFLC) submits the records of this process, which it assisted its client, the Free Software Foundation, to design and execute. [1]

1 While this 19 month transnational consultation process operated entirely on Free Software, the procedures of this Parliament require the use of proprietary document production tools and formats in order to discuss it on the public record. This document is the closest approximation to those formats that can be produced using internationally recognized standard formats and Free Software document production tools that are available to all

1 THE GPL AND COPYLEFT

The GPL is the world's most widely used Free Software licence.[2] The Free Software Foundation, the founders of the Free Software movement, defines free software as:

> [S]oftware that respects users' freedom and community. Roughly, **the users have the freedom to run, copy, distribute, study, change and improve the software**. With these freedoms, the users (both individually and collectively) control the program and what it does for them.[3]

The GPL preserves these freedoms for its users through a series of requirements in the licence. Originally designed for use in the GNU project to build a fully free computer operating system, these licence requirements are collectively referred to as “Copyleft”. The GNU project's explanation of Copyleft follows.

1.1. What is Copyleft?[4]

Copyleft[5] is a general method for making a program free software and requiring all modified and extended versions of the program to be free software as well.

The simplest way to make a program free is to put it in the public domain,[6] uncopyrighted. This allows people to share the program and their improvements, if they are so minded. But it also allows uncooperative people to convert the program into proprietary software.[7] They can make changes, many or few, and distribute the result as a proprietary product. People who receive the program in that modified form do not have the freedom that the original author gave them; the middleman has stripped it away.

In the GNU project,[8] the aim is to give *all* users the freedom to redistribute and change GNU software. If middlemen could strip off the freedom, there might be many users, but those users would not have freedom. So, instead of putting GNU software in the public domain, it has been “copylefted”. Copyleft says that anyone who redistributes the software, with or without changes, must pass along the freedom to further copy and change it. Copyleft guarantees that every user has freedom.

Copyleft also provides an incentive[9] for other programmers to add to free software. Important free programs (such as the GNU C++ compiler) exist only because of this.

Copyleft also helps programmers who want to contribute improvements[10] to free software[11] get permission to do that. These programmers often work for companies or universities that would do almost anything to get more money. A programmer may want to contribute his/her changes to the community, but her employer may want to turn the changes into a proprietary software product.

When the employer is told that it is illegal to distribute the improved version except as free software, the employer usually decides to release it as free software rather than throw it away.

To copyleft a program, it is first stated that it is copyrighted; then, distribution terms are added, which are a legal instrument that gives everyone the rights to use, modify, and redistribute the program's code *or any program derived from it* but only if the distribution terms are unchanged. Thus, the code and the freedoms become legally inseparable.

EU citizens. The requirement to use proprietary fonts, formats and tools in discussing EU free and open source software policy is a testament to the incoherence of that policy.

2 http://osrc.blackducksoftware.com/data/licenses/.

3 Emphasis original, from The Free Software definition - https://www.gnu.org/philosophy/free-sw.html.

4 Text of this section is taken almost verbatim from the Free Software Foundations' licence description text, which is available at https://www.gnu.org/licenses/.

5 See https://www.gnu.org/copyleft/copyleft.html.

6 See https://www.gnu.org/philosophy/categories.html#PublicDomainSoftware.

7 See https://www.gnu.org/philosophy/categories.html#ProprietarySoftware.

8 See https://www.gnu.org/gnu/thegnuproject.html.

9 See https://www.gnu.org/philosophy/pragmatic.html.

10 See https://www.gnu.org/software/software.html#HelpWriteSoftware.

11 See https://www.gnu.org/philosophy/free-sw.html.

Proprietary software developers use copyright to take away the users' freedom; the GNU Project uses copyright to guarantee their freedom. That's why the name is reversed, changing “copyright” into “copyleft”.

Copyleft is a general concept; there are many ways to fill in the details. In the GNU Project, the specific distribution terms that are used are contained in the GNU General Public License, the GNU Lesser General Public License (LGPL) and the GNU Free Documentation License (FDL).

The appropriate license is included in many manuals and in each GNU source code distribution.

The GNU GPL is designed so that it can easily be applied to any program by the copyright holder. The copyright holder doesn't have to modify the GNU GPL to do this, but just to add notices to the program which refer properly to the GNU GPL. However, if someone wishes to use the GPL, he/she must use its entire text: the GPL is an integral whole, and partial copies are not permitted. The same applies to the LGPL, Affero GPL, and FDL.

Using the same distribution terms for many different programs makes it easy to copy code between various different programs. Since they all have the same distribution terms, there is no need to think about whether the terms are compatible. The Lesser GPL includes a provision that allows altering the distribution terms to the ordinary GPL, so that one can copy code into another program covered by the GPL.

2 CREATING VERSION THREE OF THE GNU GENERAL PUBLIC LICENSE (GPLV3)

On January 16th, 2006 the GPL version 3 revision process began with a conference at the Massachusetts Institute of Technology. With approximately 350 participants, including 87 invited delegates serving on one of four discussion committees, this conference served as the public introduction to what would become a nearly 19 month consultation process designed to include every stake holder in one of the most widely used software licenses in the world.

2.1 The GPLv2

In January 2006, GPL version 2 was one of the most widely used software licenses in the world, a legal document tying together individuals, groups, governments, and private institutions on every continent. When GPLv2, the first version to achieve widespread adoption, was originally released in June 1991, Free Software was a small movement geographically centered around the Massachusetts Institute of Technology. In the nearly 15 years since that event, Free Software had grown by orders of magnitude, taking its place as a pillar of both business and non-commercial computer usage. The changing software landscape posed challenges for the 15 year old license. In the intervening years, software patents had become a reality in the United States, DRM technologies[12] and anti-circumvention laws were creating new restrictions on computer users’ freedoms, software licensed under the GPL had spread to a multitude of different legal jurisdictions, and new Free Software licenses had been written with provisions that made them technically incompatible with the GPL even where the communities using both licenses wished to cooperate. Change was needed to address these issues but rewriting the license by itself would have little effect. The GPL itself is not a law and all participants in the community join voluntarily. Changing the legal norms of that community would require a large process of outreach, discussion, and listening to ensure that the final terms of the new license would be not just acceptable but attractive to all members. After six months of planning, the Free Software Foundation and the Software Freedom Law Center launched the GPLv3 revision campaign to do just that.

12 “Digital Restrictions Management” or “Digital Rights Management” tools, known more commonly as “DRM,” are access control technologies that seek to dictate what an individual may do with digital content.

2.2 The Process Definition

From the beginning the GPLv3 revision process was designed to be inclusive and transparent. As such, it began with the release of a Process Definition document[13] outlining the structure of the revision process. This listed how many drafts were planned, the estimated time frame for their release, what information would be released about the reasoning behind any changes to the license at each stage, how to participate in the process, how that participation would be incorporated in writing new versions, and FSF's guiding principles in revising the license. While the final version of this 22 page document was released on January 15, 2006, just before the first international conference, early versions had been available to the public for six weeks prior to that date. Even in defining the process FSF wished to listen to the community. The final process definition outlined three main avenues for public participation: commenting on the public website, attending one of the international conferences, or participating on one of four discussion committees.

3 THE PUBLIC CONSULTATION

3.1 The Website: Stet

In order to enable direct participation in changing the text of the GPL, and do so on a large scale, the FSF commissioned the construction of custom software named "Stet". Stet's goal was to enable transparent commenting on versions of the license text as they were released. This required both the ability to easily make comments, either through the web or via email, and the ability to see what portions of the text others had commented on. At the time, this kind of collaborative commenting system was completely novel. After the successful completion of the GPL revision process, a number of government representatives contacted SFLC and FSF about adopting Stet for use in public discussions of pending legislation. FSF released Stet as free software under the GPL, and it has even been improved upon and enhanced into the "co-ment"[14] system by Phillip Aigrain's Paris-based firm Sopinspace.[15]

As discussed in the comment system documentation,[16] every effort was made to ensure that public discussion would remain productive. This was accomplished through a focus on diplomacy and public engagement at all times and by requiring that each comment be tied to specific language in the draft or language that should be inserted into the draft rather than opening the door to demands and opinions disconnected from license text. As a result, and despite sometimes heated tempers during the course of the 19 month process, the public comments remained productive without any moderation.

In total, 2,635 comments were made over the course of the revision process. All four drafts of the GPLv3 are still available with their public comments visible. As explained in the documentation, areas of the text with highlighting indicate areas with corresponding comments. The color of the highlight indicates the volume of the comments on that section, with yellow as the lowest volume of comments and red as the highest volume. In order to view the comments associated with a particular highlighted section, one simply needs to click on the text and the comments will load on the screen to the right of the license text.

- Draft 1, with 967 comments[17]
- Draft 2, with 727 comments[18]
- Draft 3, with 649 comments[19]
- Draft 4, with 292 comments[20]

13 http://gplv3.fsf.org/original-process-definition/
14 http://www.co-ment.com/
15 http://www.sopinspace.com
16 http://gplv3.fsf.org/wiki/index.php/Comment_system
17 http://gplv3.fsf.org/comments/gplv3-draft-1
18 http://gplv3.fsf.org/comments/gplv3-draft-2
19 http://gplv3.fsf.org/comments/gplv3-draft-3

3.2 Public events

For those who could not or did not wish to participate in the license revision process online, a series of conferences and community events were organized where individuals could discuss their concerns and ideas directly with representatives from the FSF and SFLC. In total 18 events were held in a dozen countries. These events included five conferences organized specifically for the discussion of the GPL revision:

- January 16th, 2006. Boston
- April 21st, 2006. Porto Alegre
- June 22nd and 23rd, 2006. Barcelona
- August 23rd, 2006. Bangalore
- November 21st and 22nd, 2006. Tokyo

To expand the discussion further, community organizations and conference organizers were encouraged to put together sessions at related conferences:

- February 10th, 2006. Bologna, Italy: Incontro al Master in Tecnologie del Software Libero
- March 18th, 2006. Torino, Italy: GPLv3 presented by Richard Stallman
- May 12th, 2006. Milano, Italy: Giornata di studio sul TCPA
- May 29th, 2006. Manchester, UK
- August 29nd, 2006. Dataföreningen Region West, Sweden
- August 30th, 2006. Copenhagen, Denmark: “Do you know enough about GPLv3?”
- September 6th, 2006. Oruro, Bolivia: VI Congress on Free software
- September 9th, 2006. Pisa, Italy: Lesson at Master for management of Free Software
- September 15th, 2006. Berlin, Germany: GPLv3 workshop at WOS4
- September 26th, 2006. Dublin, Ireland: GPL: What can v3 improve?
- October 13-15, 2006. Mendoza. Argentina
- November 4th, 2006. Dublin, Ireland: GPLv3, DRM, and the Linux kernel
- April 1, 2007. Brussels, Belgium: GPLv3 - Improving a Great Licence

Further details and event records for all these GPLv3 related meetings are available from http://gplv3.fsf.org/wiki/index.php/Event_Planning

3.3 Discussion Committees

In addition to these methods of encouraging individual participation, four discussion committees were formed to give representatives of the different groups with a stake in the license a forum for expressing the concerns of their communities and coordinating with each other. Members of these committees were asked to both represent their particular communities and actively seek out and engage other members of those communities so that everyone with a concern about the license would have a voice.

The committees were loosely organized into individual users and developers (Committee D)[21], commercial distributors and users (Committee B)[22], non-profit distributors and public or private institutional users (Committee C)[23], and representatives of international communities and large free software projects using non-GPL licenses (Committee A)[24]. In

20 http://gplv3.fsf.org/comments/gplv3-draft-4
21 Committee D materials - http://gplv3.fsf.org/discussion-committees/D/members
22 Committee B materials - http://gplv3.fsf.org/discussion-committees/B/memberlist
23 Committee C materials - http://gplv3.fsf.org/discussion-committees/C/memberlist-public
24 Committee A materials - http://gplv3.fsf.org/discussion-committees/A/committee-A-bios

total 87 representatives from these different communities were invited to form the committees with each committee also given the ability to add what other members they saw fit.

Each committee had a representative from the Free Software Foundation participate during meetings in order to help ensure that the discussions there were taken into consideration in preparing the next draft of the GPL. Each committee was given control of how and when they would meet and how much of their discussions they would make public. While Committee D choose to meet in public Internet Relay Chat (IRC) rooms and on a publicly archived mailing list, Committee B met mostly in person or on the phone and opted to keep their discussions confidential until six months after the license's release. Committees A and C opted for less formal rules. Many of these discussion materials are still available today, including the full minutes from Committee B[25] and both the IRC[26] and mail[27] records from Committee D.

In total these committees met for 80 or more hours during the course of the revision process.

4 THE DRAFTS

In total, four discussion drafts were promulgated by the FSF, though the initial process document had only anticipated three. The need for an additional draft was recognized when Microsoft and Novell announced their joint patent agreement on November 2, 2006[28] which contained discriminatory promises of patent safety[29].

Each discussion draft was accompanied by a rationale document. These documents contained the details of changes made since the previous version along with detailed reasons for each modification. The first rationale presented the FSF's goals in beginning the GPL revision process and an introduction to the modifications made since the GPLv2. Each subsequent rationale took the form of a strike-through version of the license highlighting the changes made between draft versions and footnotes explaining the reasons for each modification.

- 1st discussion draft: http://gplv3.fsf.org/gpl-draft-2006-01-16.html
  - side by side comparison between GPLv2 and GPLv3-draft1: http://www.groklaw.net/articlebasic.php?story=20060118155841115
  - rationale: http://gplv3.fsf.org/gpl-rationale-2006-01-16.html
  - Transcript of presentations at GPLv3 launch conference on January 16th, 2006: http://www.ifso.ie/documents/gplv3-launch-2006-01-16.html
- 2nd draft: http://gplv3.fsf.org/gpl-draft-2006-07-27.html
  - rationale (*pdf*): http://gplv3.fsf.org/gpl3-dd1to2-markup-rationale.pdf
- 3rd draft: http://gplv3.fsf.org/gpl-draft-2007-03-28.html
  - rationale (*pdf*): http://gplv3.fsf.org/gpl3-dd3-rationale.pdf
  - FAQ: http://gplv3.fsf.org/dd3-faq
- 4th draft (final call): http://gplv3.fsf.org/gpl-draft-2007-05-31.html
  - rationale (*pdf*): http://gplv3.fsf.org/gpl3-dd4-rationale.pdf
- Final GPL text: http://www.gnu.org/licenses/gpl-3.0.html

25 Committee B meeting minutes - http://gplv3.fsf.org/discussion-committees/B/Minutes/
26 Committee D IRC meeting minutes - http://gplv3.fsf.org/discussion-committees/D
27 Committee D mailing list archive - http://gplv3.fsf.org/pipermail/committee-d/
28 https://www.fsf.org/licensing/2007-03-28-gplv3-grandfather
29 https://www.fsf.org/news/microsoft_response

- rationale (*pdf*): http://www.gnu.org/licenses/gpl3-final-rationale.pdf
- Announcement video: http://gplv3.fsf.org/static/release/rms_gplv3_launch_high_quality.ogg
  - Transcript of announcement video: http://gplv3.fsf.org/rms_gplv3_launch_transcript
- FSF's "Quick Guide to GPLv3" about final license: https://www.gnu.org/licenses/quick-guide-gplv3.html

CONCLUSION

The GPLv3 process, documented in the foregoing materials, shows how highly specialized and economically sensitive law-making can be undertaken in a non-hierarchical and cooperative fashion, allowing individuals and powerful commercial organizations equal opportunities for participation. FOSS licensing can and should be done, as most forms of transitional regulation should be achieved, in multilateral cooperative processes.

The European Commission was invited to participate in the making of GPLv3. A representative of DGInfso attended the initial international conference at MIT on January 16. 2006, and was invited to join Discussion Committee B. The Commission declined to participate, on the ground that it could only participate in government-to-government processes, and although other governments (the Commonwealth of Massachusetts, for example) were participating, apparently they were not governments of the Commission's level of dignity and importance. It seems appropriate, on the present occasion, to consider these events.

Developing an EU model: the EUPL license

Patrice-Emmanuel Schmitz, Developer of the EUPL

ABSTRACT

The European Union Public Licence (EUPL) is a free or open source software licence, copyrighted by the European Union. It has been drafted by the Commission (under the IDAbc and ISA programmes) as from 2005 and launched in January 2007. Everyone can use it and at the end of 2012, about 500 projects were licensed under the EUPL. The present note analyses the legal aspects of the new version 1.2 of the EUPL, elaborated in 2013.

CONTENT

EXECUTIVE SUMMARY

The EUPL is the European Union Public Licence, published by the European Commission (EC). It has been studied and drafted as from 2005 and launched in January 2007. Until June 2013, the sole working version was the multilingual EUPL v1.1 (January 2009). In 2012, it was used for more than 500 software and non-software projects across Europe.

The EUPL is a Free/Open Source Software (FOSS) licence. The immediate objective thereof was that software produced under the IDA/IDABC/ISA programmes could be licensed by the EC, in a way it could be reused, improved and integrated by any recipient. The long term strategy is to bring more licensors (mainly from public sector) to follow this example. The EUPL is also a share-alike (or “copyleft”)[30] licence resulting from the aim to avoid exclusive appropriation of the covered software. The EUPL is a share-alike on both source and object code. The EUPL can be used by everyone: European institutions, Member States, economic operators and individuals.

In 2009, the EUPL v1.1 was certified by the leading open source organisation, the OSI (Open Source Initiative) as the first and sole “OSI-approved licence” with multilingual working value (in 22 languages of the European Union).

[30] « Copyleft » or « Share Alike » is the obligation (i.e. in the GPL family of licences, the EUPL, the OSL) to reuse the same licence when redistributing a covered software A, or a derivative of A.

As from 2012, the EC objective is to reinforce its legal tools (including the EUPL) for more sharing, reuse and interoperability. If "copyleft" aims to protect against appropriation, licence conflicts may also create legal barriers between FOSS communities. Therefore the EUPL includes an appendix of "compatible licences" providing interoperability with a list of similar licences. As this list (based on a 2006 study) was outdated, and to consider changes in the European legal framework a new release of the EUPL (v 1.2) was drafted and submitted to public consultation.

Modifications introduced by the EUPL v1.2 are limited : official EU institutions denominations were adapted according to the Lisbon Treaty, provisions are about "the Work" (in more general terms), additional agreements may cover a larger scope than just services and warranty (i.e. jurisdiction, venue) and interoperability is extended to new licences: GPL v3, AGPL v3, LGPL, MPL v2.

The publication of v1.2 has no impact when software was expressly covered "by the EUPL v1.1 only" (current licensors may opt for updating, or not), but v1.2 is compatible with v1.1, which can still be used.

1. MOTIVATION AND HISTORY OF THE EUPL

KEY FINDINGS

- The lack of FOSS licences fully compatible with the European legal framework and having a working value in all the European Union languages was the main motivation for writing a new European licence.
- The decision for publishing the EUPL is the outcome of a maturation process over many years.
- Writing the EUPL was a collective work coordinated by the Commission.
- The EUPL is not used "only" as a licence, but also as a reference for public procurement (for ensuring full software distribution rights to the contracting authority).
- A new version, drafted in March 2013 and planned for publication in June or July, will improve interoperability.

1.1 EUPL requirements

From 2001 to 2005 the question was: "How to share EC software and encourage public sector to do the same"? No sharing (redistribution for reusing, adapting etc.) can be done without a distribution licence. The requirements for this licence were as follow:

1. A (software) licence granting Free (or Open Source) software freedoms;
2. Ensuring protection from exclusive software appropriation (therefore being a "share alike" or "copyleft" licence);
3. Working value in all official EU languages (no need for sworn translator in Court);
4. Checked conformity with European copyright law and terminology;
5. Coverage of "communication to the public" including Web distribution / Software as a Service - SaaS (in such case, the software is not distributed as a downloaded package or as a CD-Rom, but as an application that remote users access via Internet);
6. Clarification of applicable law and competent court, as requested by EU institutions;

7. "Case law compatible" approach of warranty and liability (a general exclusion of liability is not valid facing European courts);
8. Not too long, not too complex, comprehensive and pragmatic.

During the EUPL elaboration in 2006, no existing licence was found to correspond to at least four key requirements (N° 3, 4, 6 and 7).

Therefore, the decision of writing the EUPL was taken. The EUPL is not a « Vanity licence » (where the main motivation of the author is just to forge « its own » licence and attach its name to it): it answers to a number of real issues, starting from the fact that governments and public sector organisations in general are often legally obliged to use legal instruments with a working value in their local language (N° 3).

This point is too often misunderstood by developers, especially in the US (but not only!): English is the developers' lingua franca and they consider as strange, not to say "totally irrelevant" (not to use any stronger terms!), any request by licensors or recipients to obtain working translations in another language.

At least three additional points were also very important to clarify (N°4, 6, 7): terminology, applicable law / competent court, warranty and liability.

Other points are not unique to the EUPL, even if the coverage of SaaS is still a rarity (the GNU Affero General Public License built on the GPL v3 presents similar characteristics on this specific point). Some licences (in particular the very permissive ones, like the BSD or the MIT) are much shorter, but it is commonly acknowledged that the EUPL is concise and comprehensive compared with some other "copyleft" licences.

1.2. EUPL History

The EUPL has an already long story, responding to the question: how to distribute and share European Institutions' software?

However, the wide use of the EUPL started in 2009, when the licence was translated in 22 linguistic versions and received approval from the Open Source Initiative (OSI).

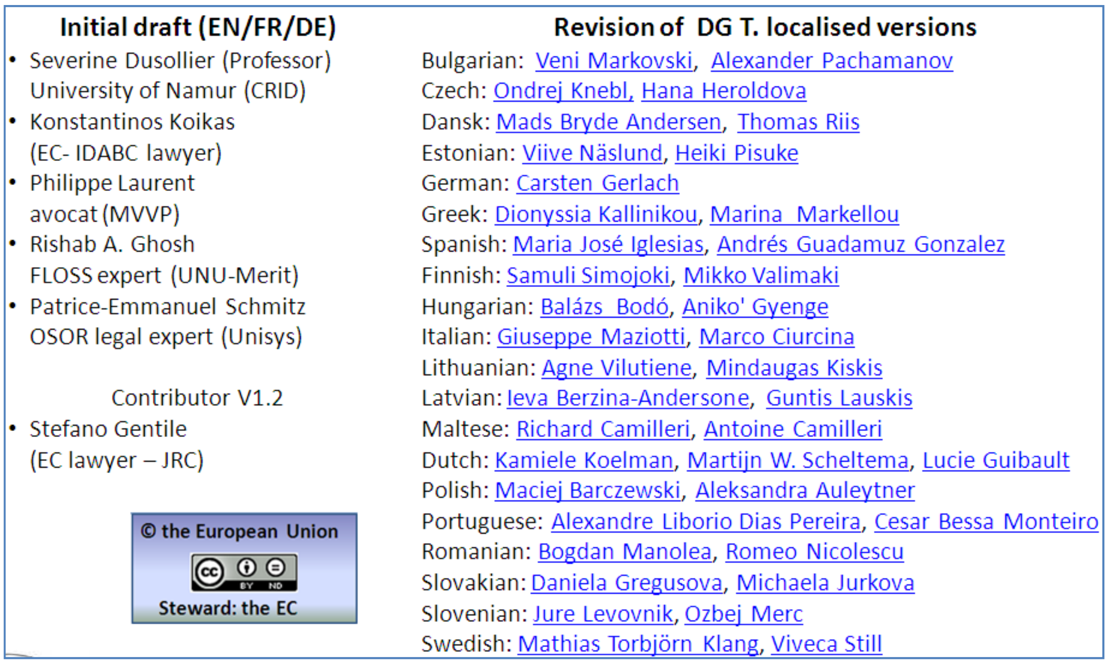
Here is a rough timetable of the EUPL history:

- 2001-2005 - "How to distribute EC software?"
- 2005 - Public consultation - Decision to create the EUPL
- 2006 - Study (CRID) for making the EUPL interoperable
- 2007 (January) - EUPL v1.0 approved by EC Commissioners
- 2008 - Elaboration of 22 linguistic « working versions »
- 2009 - EUPL v1.1 approved by EC Commissioners
- 2009 (March) - EUPL v1.1 certified by OSI.org
- by end 2012 - More than 500 projects distributed
- 2013 - Public consultation on the EUPL v1.2 (December 2012– March 2013)
- 2013 - final draft EUPL v1.2 published (probably in June or July, depending on translation and approval by the College of Commissioners).

1.3. Who wrote the EUPL?

In addition to the public consultation, which provided substantial improvements, about 50 persons contributed to the writing of the EUPL. The work from the original team was complemented by contributions from IPR lawyers from 22 Member States.

Figure 1 List of EUPL contributors



Initial draft (EN/FR/DE)

- Severine Dusollier (Professor) University of Namur (CRID)
- Konstantinos Koikas (EC- IDABC lawyer)
- Philippe Laurent avocat (MVVP)
- Rishab A. Ghosh FLOSS expert (UNU-Merit)
- Patrice-Emmanuel Schmitz OSOR legal expert (Unisys)

Contributor V1.2

- Stefano Gentile (EC lawyer – JRC)

© the European Union
Steward: the EC

Revision of DG T. localised versions

Bulgarian:	Veni Markovski, Alexander Pachamanov
Czech:	Ondrej Knebl, Hana Heroldova
Dansk:	Mads Bryde Andersen, Thomas Riis
Estonian:	Viive Näslund, Heiki Pisuke
German:	Carsten Gerlach
Greek:	Dionyssia Kallinikou, Marina Markellou
Spanish:	Maria José Iglesias, Andrés Guadamuz Gonzalez
Finnish:	Samuli Simojoki, Mikko Valimaki
Hungarian:	Balázs Bodó, Aniko' Gyenge
Italian:	Giuseppe Maziotti, Marco Ciurcina
Lithuanian:	Agne Vilutiene, Mindaugas Kiskis
Latvian:	Ieva Berzina-Andersone, Guntis Lauskis
Maltese:	Richard Camilleri, Antoine Camilleri
Dutch:	Kamiele Koelman, Martijn W. Scheltema, Lucie Guibault
Polish:	Maciej Barczewski, Aleksandra Auleytner
Portuguese:	Alexandre Liborio Dias Pereira, Cesar Bessa Monteiro
Romanian:	Bogdan Manolea, Romeo Nicolescu
Slovakian:	Daniela Gregusova, Michaela Jurkova
Slovenian:	Jure Levovnik, Ozbej Merc
Swedish:	Mathias Torbjörn Klang, Viveca Still

1.4. EUPL impact and licence proliferation

In the early days of free/open source software (until the year 2000) the GPL v2 licence and its LGPL "library" variant were adopted by some 90% of all FOSS projects.

Since then, the number and the frequency of use of other licences have increased strongly. The GPLv3 and AGPLv3 introduced in 2007 have not replaced the previous GPLv2, which still seems to be the most used licence (i.e. by Linux). Some important business projects are driven by foundations (Apache, Mozilla etc.) promoting other FOSS licences. Such licence proliferation may be considered as unfortunate, because it has made the work of developers more complex, but it looks a definitive fact: nearly every day or week, new licences are drafted, and the task of "OSI licence reviewers" seems endless.

To compensate the issue of licence proliferation, EUPL has chosen the way of interoperability (see section 4). The EUPL inspires also other governments (i.e. Quebec in Canada), which have requested to modify the EUPL for using it as template for their own needs. In such case, maintaining the same list of compatible licences may strongly reduce the impact of licence proliferation.

Similarly, the new (2013) version 2.1 of the CeCILL licence (used by French administration) includes now the EUPL and the GPL as downstream compatible licences, which looks positive for developers from both communities.

During the last months, the use of the EUPL for licensing projects was strongly growing. A November 2012 evaluation counted about 500 projects (some of them with up to 100 licensed files), and new projects are published every week. The European Parliament has selected the EUPL for the distribution of its first large FOSS project, AT4AM[31].

1.5. The EUPL used as a "reference"

Another interest of the EUPL is to be part of the European Interoperability Framework (EIF) and to be used as a reference, especially in public software requirements and procurement agreements[32]. In line with the EU ministerial declarations on the opportunity to reduce

[31] http://www.at4am.org/eupl/

[32] See the Guide for the procurement of standard-based ICT / Elements of Good Practice – (European Economics 23 March 2012) - http://cordis.europa.eu/fp7/ict/ssai/docs/study-action23/d3-guidelines-finaldraft2012-03-22.pdf and the ISA standard "Sharing and reusing clauses" http://joinup.ec.europa.eu/elibrary/document/isa_share_reuse_d_2-1-standard-sharing-and-re-using-clauses-contracts

development costs by sharing and reusing software, contracting authorities must obtain from their suppliers the right, not only to use but also preserve their rights to redistribute the developed software in the future, as the case may be (i.e. in case the development is successful, interesting for other stakeholders, and if a sharing decision is taken by the authorities).

Therefore, suppliers must not only give the "property" of the solution (including the software code), but must also grant that it can be legally distributed to third parties by the contracting authority, without any copyright issue or licence conflicts (in case several components of the solution were obtained under non-compatible FOSS licences) and royalty free (in case some proprietary standard or patents were implemented).

Example of such provision:

> *"The supplier will grant that the purchasing authority has the right to distribute the delivered application under the European Union Public Licence (EUPLv1.1 or later) or any licence(s) providing the rights stated in the article 2 of the EUPL."*

A reference to the EUPL is especially convenient due to its multi-lingual validity: it can be part of specifications written in any language of the EU.

1.6. The EUPL v1.2

The most recent evolution is the EUPL v1.2 drafted at the beginning of 2013 and is planned to be published in June or July 2013. This version is very similar to the previous v 1.1 (which can still be used), but presents the following differences:

- The terminology is adapted in consideration of the Lisbon Treaty (mainly the name of EU institutions, references to the TFEU);
- The licence covers "the Work" (which can be software, but also any other kind of copyrighted work: data, specifications, documentation etc.);
- The scope of possible "additional agreements" is enlarged (i.e. they may cover jurisdiction and any other provisions, in so far as the granted rights are not restricted);
- The list of compatible licences is extended to licences published after the initial EUPL: GNU GPLv3, AGPLv3, MPLv2 etc.

2. RIGHTS GRANTED TO RECIPIENTS BY THE EUPL

KEY FINDINGS

- Rights granted to recipients are the rights granted by all (certified) FOSS licences.
- In addition, these rights must be royalty free.

According to article 2 of the EUPL, the rights granted to the recipients of the covered software (or, under EUPL v1.2, Work) constitute a world-wide, royalty-free, non-exclusive licence to:

- use the Work in any circumstance and for all usage,
- reproduce the Work,
- modify the Original Work, and make Derivative Works,
- communicate to the public, including the right to make available or display the Work or copies thereof to the public and perform it publicly,

- distribute the Work or copies thereof,
- lend and rent the Work or copies thereof,
- sub-license rights in the Work or copies thereof.

No type of activity (i.e. commercial use) is prohibited by the EUPL: any enterprise can use the covered work for its commercial activities.

One may sell software or works covered by the EUPL and related services at a determined price (i.e. a lump sum representing a participation to the development costs of a standard or of a software, a maintenance fee for support services etc.), but once this is done, the covered work cannot be subject to the management of royalties (i.e. a fee – even small or reasonable - per use or per user). This is because the fundamental principle of FOSS is the freedom granted to all possible recipients in the world to make derivative works and to redistribute such works to anyone, making the “control” of the use and the management of royalties impossible.

Therefore, if software developers, standard developing organisations (SDO) or patent owners may cover their costs by adopting a FRAND (fair reasonable and non discriminating) licensing policy for using their work in proprietary implementations, they should also adopt a second (dual) royalty free licensing policy (like the EUPL) if they don’t want to see their standard or specification totally ignored by FOSS implementations. This would not be discriminatory against non-FOSS (or proprietary) implementations, as FOSS is not a group, a product or a technology, but a legal regime that anyone may adopt.

Concerning the use of patents, the same article 2 states that the EUPL licensor grants to the recipient of the work a royalty-free, non exclusive usage rights to any patents held by the licensor, to the extent necessary to make use of the rights granted on the work distributed under the EUPL licence.

3. WHAT MAKES THE EUPL SPECIFIC?

KEY POINTS

- The EUPL is the sole FOSS licence working in 22 languages (more will be added).
- At the contrary of other licences, the EUPL specifies an explicit warranty that contributors have copyright on their contributions.
- A single jurisdiction (the CJEU) could be requested to interpret the EUPL and copyright law in case of legal problems / litigation.
- A unique, variable "copyleft" applies, in order to ensure interoperability.

The EUPL is specific and different from all other FOSS licences on a number of points:

- Multilingualism:
  This point is the most visible: like many other European Union legal instruments, the EUPL is available in 22 languages. Gaelic and Croatian version still have to be published.
- Terminology
  The EUPL is drafted to work under European Law, even if it may be used outside the European Union and submitted to third country courts. Relevant provisions applie to the copyright terminology (the “communication to the public”), to the reasonable limitation of liability, to the reference to European treaties.

- Warranty
  The covered work is given without warranty, except one: the original licensor and every subsequent contributor grant that they are the authors (or received licence) for their own contribution. This contributes to the security of the licence (regarding possible copyright infringements) and is finally the type of requirement that you will find in all reasonable contributor agreements.
- Reference to the European Court
  Taking advantage of the treaties (TFEU) the EUPL benefits from interpretation by a unique jurisdiction: the Court of Justice of the European Union. In addition, the 28 Member States jurisdictions can address questions and be supported by a single European Court.
- Variable "Copyleft"
  The EUPL is "copyleft" on code and binaries, but this share-alike effect is, by exception for interoperability, variable[33] in the case of combined derivatives (see section 4 hereafter).
- Innovative ethic of interoperability and freedom

  These ensure that there is no exclusive appropriation of the software.

4. INTEROPERABILITY OF THE EUPL

KEY FINDINGS

- The EUPL has traced an original way to be "copyleft" and interoperable with other licences.
- This facilitates the development of other "son & grandson" projects, but has no impact on a project covered by the EUPL: there is no project relicensing.
- The notion of "strong copyleft" is still unclear and it may be that it could not be enforced in Europe.
- The EUPL approach is pragmatic, avoiding exclusive appropriation of the covered code without preventing some reuse in the framework of projects with a commercial goal.

4.1. What is legal interoperability?

Interoperability (at licence level) is the possibility to reuse the covered code in other projects, possibly in combination with code(s) covered by other licences, while keeping the freedom to distribute the resulting combination, even when considered as a derivative work under copyright law.

Interoperability is a non-issue with permissive licences (as the BSD, the MIT) because they implement no conditions for copying or merging the covered code, even inside the software code of proprietary applications.

However, interoperability is an issue when a declared objective of the licence is to keep the covered code and its evolutions under FOSS conditions, in order to avoid its exclusive appropriation.

The EUPL is a Share Alike (or "Copyleft") licence. Thus, the following question is often posed: How strong is the EUPL "copyleft"? In other words, how far must any re-distribution be done under the same EUPL licence, according to a share alike principle? And therefore,

[33] The notion of « variable copyleft » was coined for the EUPL by Rowan Wilson (Oxford University) http://www.oss-watch.ac.uk/resources/eupl

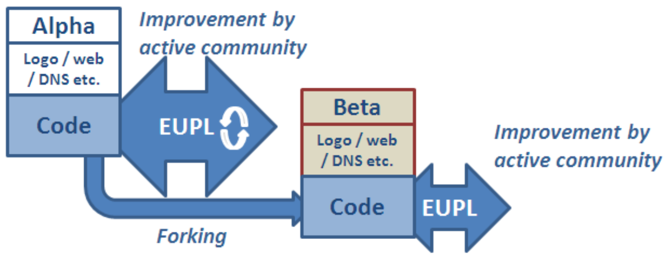
is the work protected from subsequent distribution under other licensing terms, which could lead to appropriation for the benefit of a third party software vendor?

In Europe, there are still some doubts whether "strong copyleft", whereby simply linking[34] the code covered by a "copyleft" licence with another source code automatically extends the coverage of the licence to this other source, would be generally considered lawful (in any EU member state and whatever the licence, GPL, EUPL or any other could be). There are specific exceptions for interoperability implemented by Directive 91/250 on the legal protection of computer programs. In May 2012, the Court of Justice of the European Union interpreted Directive 91/250, "as meaning that neither the functionality of a computer program nor the programming language and the format of data files used in a computer program in order to exploit certain of its functions constitute a form of expression of that program and, as such, are not protected by copyright in computer programs for the purposes of that directive"[35]. Although this judgment was not taken in the framework of free software distribution, it might have repercussions in this field, too. More particularly, it might mean that, by licensing his/her work, a copyright holder cannot prohibit the reproduction and distribution (under any other licensing terms, FOSS or non-FOSS) of the specific portions of the code that are strictly necessary for linking / implementing interoperability between the licensed program and other works, that is the data formats or APIs (application programming interfaces). Hopefully the Court will have the chance to clarify this matter in future case-law.

4.2. The normal case

Under the abovementioned reservations, we can state that the EUPL "copyleft" is as strong as possible, on code and binaries of copies and all derivative works, with defined interoperability exceptions. Let's first consider the normal case with regard to the distribution of the code (although a project is not only the code, but also other important assets (brand name, logo, site, DNS etc.)):

Figure 2: Derivative – the normal case



- A project **"ALPHA"** is more that just its software code: it is an organisation, owned by a person or a body, with an active community of developers, a web site, DNS, logo etc. Globally, this project **"ALPHA"**, can never be "re-licensed" outside the will of its original licensor (who is free, as the 100% copyright owner, to provide

[34] **Linking** makes two software working in a single application without merging their source code.

- o **Static linking** combines components through compilation, copying them into the target application and producing a merged object file that is a stand-alone executable.
- o **Dynamic linking** combines components at the time the application is loaded (load time) or during execution (run time).

[35] http://curia.europa.eu/juris/document/document.jsf?text=&docid=122362&pageIndex=0&doclang=en&mode=req&dir=&occ=first&part=1&cid=564907

exceptions or to distribute the software under various licences, called dual or multiple licensing).

- Re-distribution of the code of a project **"ALPHA"** covered by the EUPL is possible inside another project (i.e. "**BETA**", possibly known as a "forking", with another owner, brand name, logo, web site etc.), and it must be done under the same EUPL licence.

Such forking, as described in the latter scenario, is rare, at least when the original licensor organises an active community around its ALPHA project. If this is the case, all improvements will be done on ALPHA without any code re-licensing.

By exception, forking may occur for 1) licensing / philosophical reasons or 2) for functional/technical reasons:

1) A first example, is the case where the ALPHA licensor has lost its independence (i.e. is purchased by a proprietary vendor), and the community decides to re-launch to preserve EUPL licensing (not likely to happen if the licensor is a public sector body) ;
2) A second example is the case where the ALPHA licensor does not want to integrate/support new functions. For example, the Indian government wants to localise/adapt software distributed by the European Parliament in local Indian languages, but the EP does not want to be involved in this process. However, the new Indian project must also be distributed under the EUPL. The hypothesis where a significant portion of the covered code is merged in another project is similar: as a derivative, this project must be covered by the EUPL, in case it is distributed.

Once again, let’s underline the importance of an active supporting community: no forking will be sustainable in the long term without such a support.

4.3. Exception to the “normal copyleft”

The third paragraph of Article 5 of the EUPL reads as follows:

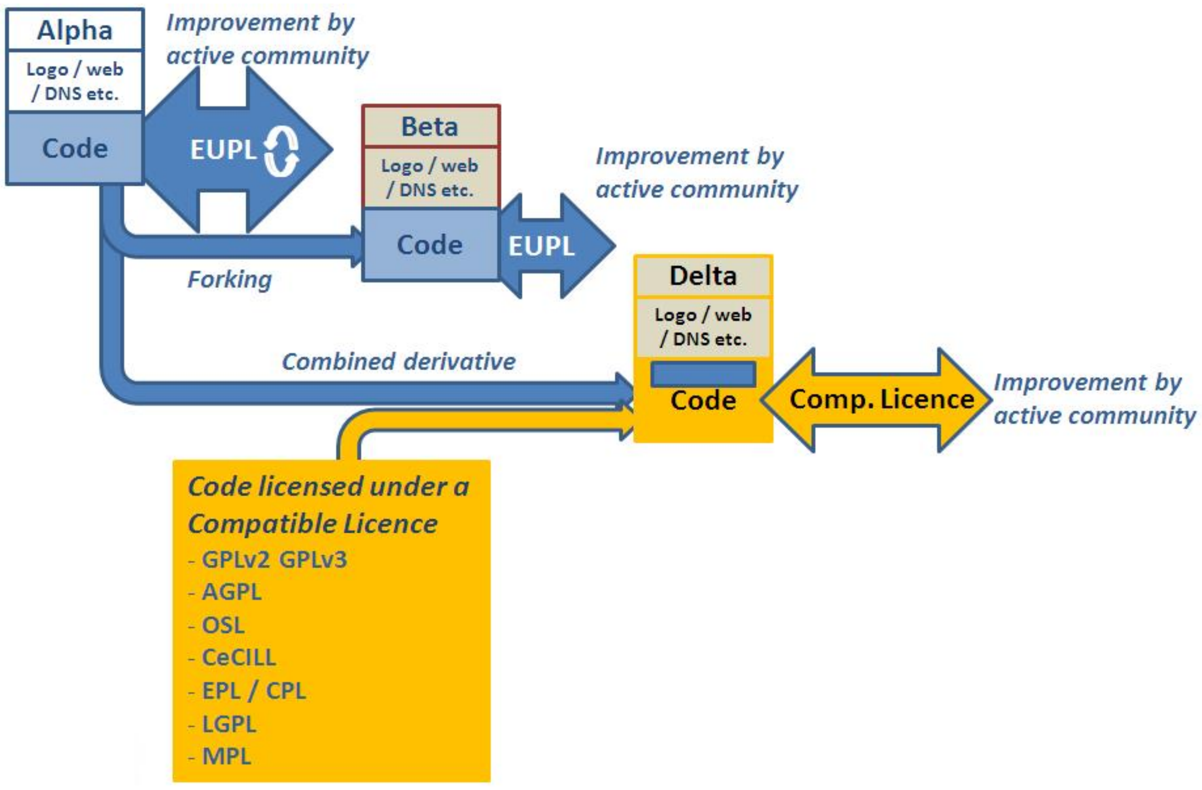
> *“If the **Licensee** Distributes and/or Communicates **Derivative Works** or copies thereof based upon **both the Original Work and another work licensed under a Compatible Licence**, this Distribution and/or Communication can be done under the terms of **this** Compatible Licence.”*

In the EUPL v1.2, a list of compatible licences is published in Appendix and contains the following names:

- *GNU General Public License (GPL) v. 2, v. 3*
- *GNU Affero General Public License (AGPL) v. 3*
- *Open Software License (OSL) v. 2.1, v. 3.0*
- *Eclipse Public License (EPL) v. 1.0*
- *CeCILL v. 2.0, v. 2.1*
- *Mozilla Public Licence (MPL) v. 2*
- *GNU Lesser General Public Licence (LGPL) v. 2.1, v. 3*
- *Creative Commons Attribution-ShareAlike v. 3.0 Unported* (CC BY-SA 3.0) - for works other than software
- *European Union Public Licence (EUPL), any version as from 1.1*

The interoperability exception will allow recipients to launch a new project DELTA, to reuse files or source code covered by one of the above licences in the DELTA project, to insert or merge the EUPL covered code in DELTA and to licence DELTA as a whole under this compatible licence.

Figure 3 Exception for compatible licences



Conditions for such « variable copyleft » are as follows:

1. Software code covered by the EUPL is combined in/with another, different work.
2. The combination (larger work) forms a derivative. Merged code must be licensed globally as a whole. Keeping distinct licences (like for the various parts of an aggregate) is not possible.
3. The other work, in which the code covered by the EUPL is merged, had been obtained under a compatible licence (according to the list).
4. The same compatible licence (according to the list) is used to license the new larger work "as a whole".

The exception for compatible licence described above should not be understood as the possibility to "relicense" a project[36]. As said above, this is obviously not the case: the reuse of some code in the project "DELTA" will not impact the project "ALPHA".

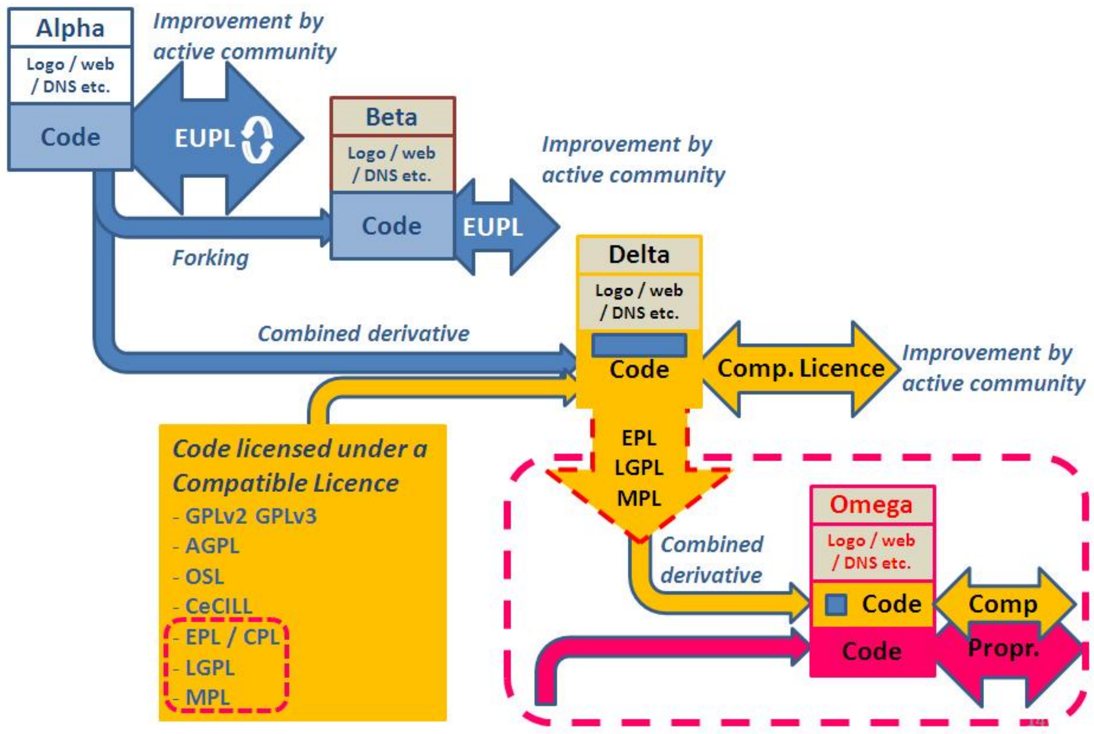
Is there any risk to see someone licensing some trivial code (like "hello world") under a compatible licence for creating a "formal larger work" and licensing it under this compatible licence? No cases were reported in five years EUPL distribution (2007-2012). It is not the way FOSS operates. Making trivial forking is losing time and reputation. A forked work is sustainable only when a working community takes it over and improves it substantially.

4.4. Exception to the exception

Because three of the listed compatible licences are more moderately "copyleft" (or only at file level) it may be that some code covered by the EUPL could also be reused in a third generation project covered – in binary executable form - by a non-FOSS licence.

[36] For example, the too brief formulation used by the Free Software Foundation may induce recipients in error : « *The EUPL allows relicensing to GPLv2, because that is listed as one of the alternative licenses that users may convert to* » http://www.gnu.org/licenses/license-list.en.html#GPLIncompatibleLicenses

Figure 4 Exception to the exception



It is therefore possible that the "daughter project" DELTA:

1. will be (by decision of its licensor and because code under these licences (EPL, LGPL or MPL) was reused) distributed under one of the more moderately copyleft licences listed as compatible (EPL, LGPL or MPL);
2. that some code from DELTA will be combined or forked in a "grand-daughter" project OMEGA, and;
3. that the OMEGA licensor will decide to distribute its executable version under proprietary terms.

Even in such a case (that has never occurred in real world so far) the portions of the DELTA code present in OMEGA will stay covered by their licence (EPL, LGPL or MPL): these files must stay FOSS and publicly available as source code, but the copyleft is limited at file level according to the provision of these licences (meaning without a pretention for viral effect impacting the rest of the OMEGA project).

This possible exception has made some analysts to declare that the EUPL "gives recipients ways to relicense the work under the terms of other selected licenses, and some of those only provide a weaker copyleft. Thus, developers can't rely on this license to provide a strong copyleft"[37].

This point is of course – at least theoretically – founded. But we have to see it in a context, and temper it:

- the term "relicense" is especially ambiguous and not appropriate, as previously stated.
- Some compatible licences provide a weaker "copyleft" (LGPL, MPL, EPL): it does not mean that they are weak or permissive: they are "copyleft", but at file level, without pretention to provide a viral effect.
- Providing a "strong copyleft" is not a business ideal in an interoperable world, where multiple licences coexist. Furthermore, the notion of "strong copyleft" is especially unclear, debated and has not been confirmed by European case law.

[37] FSF – op. cit.

4.5. Conclusion

The above series of example illustrate that the EUPL

- protects effectively the covered code from exclusive appropriation by a third party;
- make some part the covered code reusable in OTHER free software projects (without re-licensing the original project);
- is not hostile/does not try to prevent the reuse of some code of these other projects by the software industry.

REFERENCES

- The EUPL v1.1 – text of the licence, in 22 languages – Joinup.eu. https://joinup.ec.europa.eu/software/page/eupl
- Guidelines for using the EUPL https://joinup.ec.europa.eu/software/page/eupl/eupl-guidelines
- Guidelines on public procurement of Open Source Software https://joinup.ec.europa.eu/elibrary/document/guideline-public-procurement-open-source-software
- Spanish Royal decree 4/2010 (English version) see in particular article 16 http://administracionelectronica.gob.es/recursos/pae_000002017.pdf
- The EUPL in Italy: www.eupl.it
- "Experience of *introducing* the EUPL in ISTAT" (Carlo Vacari 2010) presentation slides (in Italian) : http://fr.slideshare.net/vaccaricarlo/introduzione-eupl-in-istat
- Malta public sector software distribution policy https://www.mita.gov.mt/MediaCenter/PDFs/1_GMICT_P_0097_Open_Source_Software_v2.0.pdf
- Guide for the procurement of standard-based ICT / Elements of Good Practice – (European Economics 23 March 2012) - http://cordis.europa.eu/fp7/ict/ssai/docs/study-action23/d3-guidelines-finaldraft2012-03-22.pdf
- ISA standard "Sharing and reusing clauses" http://joinup.ec.europa.eu/elibrary/document/isa_share_reuse_d_2-1-standard-sharing-and-re-using-clauses-contracts

ANNEX: TEXT OF THE EUPL (V1.2 – ENGLISH VERSION)

European Union Public Licence V. 1.2

EUPL © the European Union 2007, 2013

This European Union Public Licence (the "EUPL") applies to the Work (as defined below) which is provided under the terms of this Licence. Any use of the Work, other than as authorised under this Licence is prohibited (to the extent such use is covered by a right of the copyright holder of the Work).

The Original Work is provided under the terms of this Licence when the Licensor (as defined below) has placed the following notice immediately following the copyright notice for the Original Work:

Licensed under the EUPL

or has expressed by any other means his willingness to license under the EUPL.

1. Definitions

In this Licence, the following terms have the following meaning:

- *The Licence*: this Licence.

- *The Original Work:* the work or software distributed and/or communicated by the Licensor under this Licence, available as Source Code and also as Executable Code as the case may be.

- *Derivative Works*: the works or software that could be created by the Licensee, based upon the Original Work or modifications thereof. This Licence does not define the extent of modification or dependence on the Original Work required in order to classify a work as a Derivative Work; this extent is determined by copyright law applicable in the country mentioned in Article 15.

- *The Work*: the Original Work and/or its Derivative Works.

- *The Source Code*: the human-readable form of the Work which is the most convenient for people to study and modify.

- *The Executable Code:* any code which has generally been compiled and which is meant to be interpreted by a computer as a program.

- *The Licensor*: the natural or legal person that distributes and/or communicates the Work under the Licence.

- *Contributor(s):* any natural or legal person who modifies the Work under the Licence, or otherwise contributes to the creation of a Derivative Work.

- *The Licensee* or *"You":* any natural or legal person who makes any usage of the Work under the terms of the Licence.

- *Distribution* and/or *Communication*: any act of selling, giving, lending, renting, distributing, communicating, transmitting, or otherwise making available, on-line or off-line, copies of the Work or providing access to its essential functionalities at the disposal of any other natural or legal person.

2. Scope of the rights granted by the Licence

The Licensor hereby grants You a world-wide, royalty-free, non-exclusive, sub-licensable licence to do the following, for the duration of copyright vested in the Original Work:

- use the Work in any circumstance and for all usage,
- reproduce the Work,
- modify the Original Work, and make Derivative Works based upon the Work,
- communicate to the public, including the right to make available or display the Work or copies thereof to the public and perform publicly, as the case may be, the Work,
- distribute the Work or copies thereof,
- lend and rent the Work or copies thereof,
- sub-license rights in the Work or copies thereof.

Those rights can be exercised on any media, supports and formats, whether now known or later invented, as far as the applicable law permits so.

In the countries where moral rights apply, the Licensor waives his right to exercise his moral right to the extent allowed by law in order to make effective the licence of the economic rights here above listed.

The Licensor grants to the Licensee royalty-free, non exclusive usage rights to any patents held by the Licensor, to the extent necessary to make use of the rights granted on the Work under this Licence.

3. Communication of the Source Code

The Licensor may provide the Work either in its Source Code form, or as Executable Code. If the Work is provided as Executable Code only, the Licensor provides in addition a machine-readable copy of the Source Code of the Work along with each copy of the Work that the Licensor distributes or indicates, in a notice following the copyright notice attached to the Work, a repository where the Source Code is easily and freely accessible for as long as the Licensor continues to distribute and/or communicate the Work.

4. Limitations on copyright

Nothing in this Licence is intended to deprive the Licensee of the benefits from any exception or limitation to the exclusive rights of the rights owners in the Original Work, of the exhaustion of those rights or of other applicable limitations thereto.

5. Obligations of the Licensee

The grant of the rights mentioned above is subject to some restrictions and obligations imposed on the Licensee. Those obligations are the following:

Attribution right: the Licensee shall keep intact all copyright, patent or trademarks notices and all notices that refer to the Licence and to the disclaimer of warranties. The Licensee must include a copy of such notices and a copy of the Licence with every copy of the Work he/she distributes and/or communicates. The Licensee must cause any Derivative Work to carry prominent notices stating that the Work has been modified and the date of modification.

Copyleft clause: If the Licensee distributes or communicates copies of the Original Works or Derivative Works, this Distribution or Communication will be done under the terms of this Licence or of a later version of this Licence unless the Original Work is expressly distributed only under this version of the Licence. The Licensee (becoming Licensor) cannot offer or impose any additional terms or conditions on the Work or Derivative Work that alter or restrict the terms of the Licence.

Compatibility clause: If the Licensee Distributes or Communicates Derivative Works or copies thereof based upon both the Original Work and another work licensed under a Compatible Licence, this Distribution or Communication can be done under the terms of this Compatible Licence. For the sake of this clause, “Compatible Licence” refers to the licences listed in the appendix attached to this Licence. Should the Licensee’s obligations under the Compatible Licence conflict with his/her obligations under this Licence, the obligations of the Compatible Licence shall prevail.

Provision of Source Code: When distributing and/or communicating copies of the Work, the Licensee will provide a machine-readable copy of the Source Code or indicate a repository where this Source will be easily and freely available for as long as the Licensee continues to distribute and/or communicate the Work.

Legal Protection: This Licence does not grant permission to use the trade names, trademarks, service marks, or names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the copyright notice.

6. Chain of Authorship

The original Licensor warrants that the copyright in the Original Work granted hereunder is owned by him/her or licensed to him/her and that he/she has the power and authority to grant the Licence.

Each Contributor warrants that the copyright in the modifications he/she brings to the Work are owned by him/her or licensed to him/her and that he/she has the power and authority to grant the Licence.

Each time You accept the Licence, the original Licensor and subsequent Contributors grant You a licence to their contributions to the Work, under the terms of this Licence.

7. Disclaimer of Warranty

The Work is a work in progress, which is continuously improved by numerous Contributors. It is not a finished work and may therefore contain defects or “bugs” inherent to this type of development.

For the above reason, the Work is provided under the Licence on an “as is” basis and without warranties of any kind concerning the Work, including without limitation merchantability, fitness for a particular purpose, absence of defects or errors, accuracy, non-infringement of intellectual property rights other than copyright as stated in Article 6 of this Licence.

This disclaimer of warranty is an essential part of the Licence and a condition for the grant of any rights to the Work.

8. Disclaimer of Liability

Except in the cases of wilful misconduct or damages directly caused to natural persons, the Licensor will in no event be liable for any direct or indirect, material or moral, damages of any kind, arising out of the Licence or of the use of the Work, including without limitation, damages for loss of goodwill, work stoppage, computer failure or malfunction, loss of data or any commercial damage, even if the Licensor has been advised of the possibility of such damage. However, the Licensor will be liable under statutory product liability laws as far such laws apply to the Work.

9. Additional agreements

While distributing the Original Work or Derivative Works, You may choose to conclude an additional agreement, defining obligations and/or services consistent with this Licence. However, if accepting obligations, You may act only on your own behalf and on your sole responsibility, not on behalf of the original Licensor or any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against such Contributor by the fact You have accepted any warranty or additional liability.

10. Acceptance of the Licence

The provisions of this Licence can be accepted by clicking on an icon “I agree” placed under the bottom of a window displaying the text of this Licence or by affirming consent in any other similar way, in accordance with the rules of applicable law. Clicking on that icon indicates your clear and irrevocable acceptance of this Licence and all of its terms and conditions.

Similarly, you irrevocably accept this Licence and all of its terms and conditions by exercising any rights granted to You by Article 2 of this Licence, such as the use of the Work, the creation by You of a Derivative Work or the Distribution or Communication by You of the Work or copies thereof.

11. Information to the public

In case of any Distribution and/or Communication of the Work by means of electronic communication by You (for example, by offering to download the Work from a remote location) the distribution channel or media (for example, a website) must at least provide to the public the information requested by the applicable law regarding the Licensor, the Licence and the way it may be accessible, concluded, stored and reproduced by the Licensee.

12. Termination of the Licence

The Licence and the rights granted hereunder will terminate automatically upon any breach by the Licensee of the terms of the Licence.

Such a termination will not terminate the licences of any person who has received the Work from the Licensee under the Licence, provided such persons remain in full compliance with the Licence.

13. Miscellaneous

Without prejudice of Article 9 above, the Licence represents the complete agreement between the Parties as to the Work.

If any provision of the Licence is invalid or unenforceable under applicable law, this will not affect the validity or enforceability of the Licence as a whole. Such provision will be construed and/or reformed so as necessary to make it valid and enforceable.

The European Commission may publish other linguistic versions and/or new versions of this Licence and/or updated versions of the Appendix, so far this is required and reasonable, without reducing the scope of the rights granted by the Licence. New versions of the Licence will be published with a unique version number.

All linguistic versions of this Licence, approved by the European Commission, have identical value. Parties can take advantage of the linguistic version of their choice.

14. Jurisdiction

Without prejudice to specific agreement between parties,

- any litigation resulting from the interpretation of this License, arising between the European Union institutions, bodies, offices or agencies, as a Licensor, and any Licensee, will be subject to the jurisdiction of the Court of Justice of the European Union, as laid down in article 272 of the Treaty on the Functioning of the European Union,
- any litigation arising between other parties and resulting from the interpretation of this License, will be subject to the exclusive jurisdiction of the competent court where the Licensor resides or conducts its primary business.

15. Applicable Law

Without prejudice to specific agreement between parties,

- this Licence shall be governed by the law of the European Union Member State where the Licensor has his seat, resides or has his registered office,
- this licence shall be governed by Belgian law if the Licensor has no seat, residence or registered office inside a European Union Member State.

Appendix

“Compatible Licences” according to Article 5 EUPL are:

- *GNU General Public License (GPL) v. 2, v. 3*
- *GNU Affero General Public License (AGPL) v. 3*
- *Open Software License (OSL) v. 2.1, v. 3.0*
- *Eclipse Public License (EPL) v. 1.0*
- *Cecill v. 2.0, v. 2.1*
- *Mozilla Public Licence (MPL) v. 2*
- *GNU Lesser General Public Licence (LGPL) v. 2.1, v. 3*
- *Creative Commons Attribution-ShareAlike v. 3.0 Unported* (CC BY-SA 3.0) for works other than software
- *European Union Public Licence (EUPL), any version as from v. 1.1*

The European Commission may:

- update this Appendix to later versions of the above licences without producing a new version of the EUPL.
- extend this Appendix to new licences providing the rights granted in Article 2 of this Licence and protecting the covered Source Code from exclusive appropriation.

A discussion of the different software licensing regimes

Avv. Carlo Piana, Lawyer

ABSTRACT

Free Software, or Open Source Software, is ubiquitous. Having a minority share in consumer Personal Computer software, it has the lion's share in virtually all other markets like smartphones, Internet appliances, and cloud services. This work provides an outline of the legal aspects of Free Software, explaining how this licensing model can prosper while subverting a legal environment conceived to achieve its opposite. It also outlines how different legislations can be detrimental to it (mainly, by extending patent protection to software).

CONTENT

EXECUTIVE SUMMARY

Free Software,[38] also known, or perhaps best known, as "Open Source Software", is any kind of Software that, by being distributed under a **Free Software License,** benefits from the **Four Freedoms**:

Freedom #0 to use the software for any purpose;

Freedom #1 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 to redistribute copies so you can help your neighbour;

38 "Free" in "Free Software" has meaning as in "Freedom of speech", it does not relate to price (as in "Free Beer"), rather on being unrestricted. This is why, to avoid the ambiguity that the word "free" has in current English, it is frequently referred to as Free/Libre Software. "Open Source", conversely, is a more common way (although more recently coined) of referring to Free Software. Sometimes these naming conventions are mixed together, such as in "Free and Open Source Software" (FOSS) or even "Free/Libre and Open Source Software" (FLOSS). I use Free Software in the remainder of this work.

Freedom #3 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.[39]

Free software is, therefore, a characteristic attached to software distribution by means of a license. It is, in essence, a legal phenomenon. The word "free" does not relate at all to the price of software, but to the rights conveyed by the license.

This study makes a comparative analysis of the main features of the different licenses and aims at providing the general reader with sufficient knowledge to have a workable understanding of the really complex world of Free (open source) Software licensing. It does so from a European perspective, although the same concepts and rules can largely apply world-wide.

It shows how, from a legal perspective, Free Software revolves around licenses, mainly operating in the copyrighted world. It puts Free Software under a light that differs from commonly widespread views, which see it as an oddity from a legal perspective. Instead, Free Software is shown to be – historically – an exception to an exception, the first exception being the proprietary software and the initial normality being Free Software. Certainly, proprietary software has "environmental" advantages – from a legal standpoint – because all the legislation on the legal protection of software revolves around the concept of “all rights restricted”, so that to sell one copy of proprietary software one does not even need a legal instrument, whereas to the correct working of Free Software licensing is not only needed, but usually must also be very complex, detailed and sometimes very strict. Free Software lives in a “hostile” legal environment, but has proven to be quite resilient, also legally so, as Chapter 3 shows.

If Chapter 1 provides a historical description of why in mid-'80s the Free Software concept had to be re-engineered after the rise and domination of proprietary software, Chapter 2 provides the main building blocks of a legal theory of the three main genres of Free Software, which bring along very different legal consequences: copyleft, weak copyleft, non copyleft. **Copyleft** is a totally new concept that uses copyright in a very creative way: instead of using the restrictions provided by the said legal regime to obtain monetary compensations in exchange for trading the corresponding permissions, copyleft uses such restrictions to make sure that the rights granted are not taken back by the recipients; in other words, what is Free remains Free. Depending on the latitude of the copyleft conditions, or better, on its scope, very different obligations and conditions are attached to software distribution, and therefore resilience to proprietarization varies greatly.

The **European Union Public License (EUPL)** is also discussed in this framework. The discussion is particularly fit in this chapter because of the particular nature of the EUPL, which is metamorphic, in a way, given its compatibility clause that implements a concept commonly referred to as “legal interoperability” - upon which the author remains very sceptical.

Chapter 3 discusses how Free Software is a multi-dimensional space where one dimension is independent from all others. The discussion starts from a common mistake, that Free Software is a development model where the developers work independently and unorganized, perhaps from their garage, whereas this is just one possible occurrence of the many different possible conceptual models that can be useful to describe a particular example of Free Software development. In the same conceptual model (which considers the degree of sparseness, the control and governance, and the professionality of development) there are examples covering the whole spectrum, from hobbyists to large enterprises, from sparse development to very concentrated, or one-entity, development, and from very loosely coordinated projects to tightly managed ones. Other conceptual models take into consideration different characteristics, such as whether the software is backed by business entities or communities of individual developers, or again whether the software is distributed only as Free Software or there is some proprietary licensing

39 From http://www.gnu.org/philosophy/free-sw.html

accompanying it (e.g. in the form of dual licensing or "open core"). All of these conceptual models can be used to form the axes of the multi-dimensional space mentioned above.

To this effect, the Chapter also includes a discussion of dual licensing, where pure Free Software distribution lives together with proprietary exploitation of the same code – and consolidated one-entity ownership. Another business model that is discussed is "open core", which combines Free and proprietary Software, but in a totally different setting and without consolidation of copyright title. Another independent axis is considered, linked to another commonly misunderstood concept: that Free Software is all about community. Communities are an important part of the Free Software world, but reality is far more complex.

As mentioned, Chapter 4 answers the question whether – and why – Free Software licensing is valid and enforceable, and also includes a more in-depth analysis of the liability regime that comes together with software distribution. Because of the particular conditions under which Free Software is distributed, the liability regime is very likely to be opposite to that applicable to proprietary software.

Chapter 5 discusses how copyright is not the only legal area that is relevant to Free Software. Software is in a peculiar position since it has many different protections and interactions. The most relevant point in this discussion is the stark contradiction between Free Software (and software in general) and **patents**, and how Free Software reacts against patents. The Chapter also includes a short mention of a particular kind of **standards**, namely patent-ridden standards, which are at odds with Free Software, including some commonly referred to as "open standards" which are not open.

Finally, Free Software is a way of distributing software. But the most outspoken buzzword nowadays is "**Cloud**", and cloud is a form of *non distribution* of software. But cloud services as we know them would be virtually impossible, quite ironically, without Free Software. A short discussion of how this is relevant and what Free Software has conceived to face the hurdles will also be included.

1. HISTORICAL BACKGROUND

Free Software, or "open source software",[40] dates back to the early days of computer science, although only relatively recently it has become a subject of public discussion as such.

At the beginning of the computer industry, there was no point in defining "Free Software" at all, because software was not a separate subject from the hardware where it ran. Much of the programming was specifically written for the few computers that existed.[41] The main sectors where software was made were essentially three: military, university, hardware manufacturing.

Military did not distribute software at all. Software distribution occurred only in the hardware manufacturing industry, but only in the form of installed software as an accessory of the "real" computer, and in university.

1.1 The Academic licenses were the first form of Free Software

The first forms of Free Software can be found precisely in university. Academic software was originally distributed under (implied or express) conditions that allowed redistribution and modification. The most important of these conditions was to provide an attribution of

40 "Free Software" and "open source software" are two wordings that move from different premises, but for all practical purposes they can be considered synonyms and they are to be read in this work as strict synonyms. Please note that Free Software, as well as the term "Free" as in "Free speech" are written capitalized in this paper as a defined term, so that possible confusions with the homonym "free" which means "at no cost, gratis" are avoided.

41 For the quite famous quote of the IBM chairman "I think there is a world market for maybe five computers", although there is no direct evidence that it was ever uttered, see http://en.wikipedia.org/wiki/Thomas_J._Watson#Famous_misquote

the origin of the code, that is to say, to preserve the statement that claims authorship of the code in all subsequent distribution even of modified versions of it. This practice was not unlike the practice of citations in scientific works – which software was considered to be. It was commonplace to have these conditions spelled out in a text that accompanied the software distribution, or "the license". Universities used a standardized form of this license, which usually took the name of the University itself. The most known and used licenses were those coming from Berkeley ("BSD" = Berkeley Software Distribution) and from the Massachusetts Institute of Technology ("MIT").

This form of Free Software distribution is therefore often referred to as "Academic licensing", or "Attribution only".

1.2 Enter the Independent software vendors

Military, academic and hardware industry did not rely on any particular form of protection for the software that they made. Military simply did not distribute it. For the hardware industry software, it was just a necessary complement to their main product and there was little incentive to copy software that had to be drastically changed in order to run on other hardware. Academics freely distributed their software; their main concern was avoiding plagiarism and being acknowledged for the quality of what they wrote: they were seeking recognition.

But increasingly over the time software became disentangled from the hardware that it targeted. The operating system Unix[42] was an important part of this process, as it had a kernel that was designed to run the basic interaction with the hardware and a "user space" where applications could run and – to a large extent – be independent from the underlying hardware. This was also due to a fundamental advancement in the software programming techniques, with the creation of the C programming language by Dennis Ritchie,[43] to which Unix was ported as early as 1972.[44]

Thanks to these advancements – as well as to more performing hardware at lower prices where the overhead necessary to this abstraction of software from the hardware could be accommodated – industry specialization increased, and manufacturers that were concentrating only on software making were a natural evolution. This was the Independent Software Vendors (ISV) industry.

With independence and abstraction, as well as complexity and value of software, came the need to "protect" the investment in producing good software and to avoid that others (including the hardware makers who contracted out software to ISVs) could have a free run on this work. This protection was conceivably obtained through three different tools:

- secrecy;
- hardware keys or other form of *technical protection*;
- legal protection.

Hardware keys are not relevant to our discussion. It is a form of encryption of software that needs a hardware device of sort to run, so that copying the software is useless if the hardware device is not also obtained.

Secrecy and legal protection are far more relevant, and will be discussed in the next chapter.

42 UNIX is an alteration of the acronym UNICS, or UNiplexed Information and Computing Service. Hereinafter the common uncapitalized form "Unix" will be used. Unix is an operating system, that is, the part of a computer's software that provides the most basic functions used by more specialized software applications, roughly speaking taking care of the interactions with the hardware. For instance, an operating system checks the hardware environment, manages storage devices, collects the inputs of the user and provides the output to the user (e.g., through a monitor), connects with other computers through network interfaces, recognizes and manages devices, authenticates users and allows them to perform their permitted actions, etcetera.

43 http://en.wikipedia.org/wiki/C_%28programming_language%29

44 See http://en.wikipedia.org/wiki/Unix#1970s. For a more detailed recount of why UNIX is relevant to Free Software, see Meeker H.J., *The Open Source Alternative*, Wiley, New Jersey, USA, pp. 4-5

1.3 Source code, machine code and reverse engineering

The “traditional” way of making software, the paradigm of which can be seen in the C programming, is that the programmer uses some sort of programming language which, to an uneducated eye, would look like a mixture of English and mathematics notations, which is called “**source code**”. Source code is “human readable” code, in other words, an expert human being is able to read it and tell what the code is supposed to do. But this code cannot be used by computers, because the instructions that a computer needs are really much different, as computer need instructions in “**machine code**” or “executable form”. The translation of source code into machine code is called “**compilation**” and the application that makes this machine code is called “compiler”.

Machine code, unlike source code, would look like a series of hexadecimal characters (from 0 to 9 and from a to f) without an apparent structure or meaning. **Machine code is not human-readable**. In other words, it is completely opaque to the human being, but it is not so to a computer.

Therefore the ISV found it natural not to distribute the source code, so that any **modification** would need their intervention, or access to source code. Possession of source code in the common parlance is a proxy for “ownership” of the code, although legally speaking this is not true. In this way, ISV relied on secrecy to preserve their commercial power.

It is technically possible to obtain a close equivalent of the source code through **reverse engineering** techniques that are referred to as “**decompilation**” (the opposite of “compilation”). This practice was quite early considered illegal, a form of industrial espionage. Therefore secrecy also provides some sort of legal protection.

This way of protection (that is, hardware protection) cannot do anything against a one-to-one copy, i.e., against making an identical copy of the machine code that would run on an identical or compatible computer. With increasingly lower cost of copying, that practice became very convenient and ISVs felt they needed some protection against it. Leaving aside hardware protection or other technical anti-copying means, the only way was to have legal protection.

1.4 Legal protection: copyright

The initial debate as to what form of legal protection was to be given to software, if any, converged very soon and naturally towards copyright, both due to the nature of software (which is originally a work of writing) and to the need to protect it mainly against copying rather than against imitating. Another advantage of using copyright as a legal device to protect software was that copyright is universally protected under the Berne Convention that establishes a world-wide Copyright Union within which a copyrighted subject is uniformly and automatically protected by all the member states of the union.

The other options, such as using patents or some sort of *sui generis* rights, were considered impractical. In Europe, protection of software as a copyright subject was eventually harmonized through the Software Directive.[45]

1.5 The rise of proprietary and the re-birth of Free Software, thanks to a printer

ISVs could therefore benefit from a sound legal and technical environment. The commoditization of the PC platform made the software industry as relevant – if not more relevant - as the hardware industry. Software became a tradable object, a commercial product that, thanks to the protections granted to it, was made artificially scarce and therefore could have a price tag. Software and all copies thereof became a property that could be sold. Thus the name “**proprietary**”, which is used to define software protected and treated as property.

45 Directive 91/250/EEC, http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31991L0250:EN:HTML

The short and ultra-simplified history above shows that proprietary software is a relatively new economic paradigm in software production. It became very popular for a number of reasons, and it became in the common perception *the* way that software was made. Some disagreed that this success was deserved or that such a paradigm was the only, or the best, way to make software. One of the most famous critics was **Richard M. Stallman**.

Stallman was a young programmer and researcher at MIT in the early eighties. To him, acquainted with the academic way of making and sharing software, what happened one day with a printer was quite a shocking finding.[46]

The department where Stallman was working shared a networked printer. They used it to print sometimes long documents. The printer was frequently jammed, but because the jobs were long and the printer distant from the working space, it could happen that Stallman went to collect his batch only to discover that the job was delayed by someone who launched another one before without checking its result. A lot of delay ensued. Stallman reworked the source code of the printing software so that instead of just triggering a visual signal, a network message was sent to the owner of the job, who could timely go to the printer and remove the jam.

When a new ultra-fast networked laser printer was donated, Stallman thought he could do the same modifications, but to his disbelief he was not able to find the new source code. Believing it was a mistake, he asked for it to the developers who wrote the software, only to find that it was not an error, but a protection of the copyright of the manufacturer and that there was no chance to restore the previous functionality added by Stallman.

Stallman took it rather personally and decided that he would never tolerate this nonsensical prohibition against his efforts to improve the software and benefit his peers. He decided to start a new initiative, which later became the Free Software Foundation, to build an entirely new operating system and applications that run on it, all released as Free Software. The operating system was called GNU.[47] When in 1991 Linus Torvalds published Linux, a Unix kernel (the lowest level of functions of an operating system) for the i386 platform – and decided to change its license to the GNU GPL, the license under which GNU was released – GNU and Linux could be combined to make GNU/Linux, the operating system now commonly known with the “Linux” name.

Why this second wave of Free Software was different from the first, academic, one will be the argument of the following chapter.

2. COYPRIGHT, COPYLEFT, COPY-WHAT? A NEW FAMILY OF LICENSES

Software is protected by copyright. It is also protected by secret, when the corresponding source code is not distributed. This was the situation that Stallman faced. He was not allowed to change the code that controlled “his” printer, because he was lacking the legal rights (copyright also prohibits modifications of the protected work without the permission of the rightsholder). He was also unable to change it, since he lacked the technical means to do it, that is, he lacked the source code.

The goal of the Free Software Foundation is to remove these two obstacles for all the recipients of the software. Therefore the four fundamental Freedoms were conceived:

Freedom #0 to use the software for any purpose

Freedom #1 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 to redistribute copies so you can help your neighbour;

46 The whole story is narrated by Sam Williams, *Free as in Freedom. Richard Stallman's Crusade for Free Software,* March 2002. The text is available at http://oreilly.com/openbook/freedom/ch01.html
47 GNU is a recursive acronym “GNU's Not Unix”. Indeed GNU is a rewriting from scratch of a UNIX operating system.

Freedom #3 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.[48]

One problem was that these freedoms already applied to the first wave of Free Software. This did not prevent the proprietarization of much of the academic software and the birth of a large number of reciprocally incompatible versions of Unix, which initially was freely distributed and modified thanks to a very liberal grant by AT&T Bell Laboratories. In order for the new Unix and all Free Software to be a "commons" and more importantly to **remain a commons**, a simple device was conceived. The permission that accompanied the software was not unlimited and virtually unconditioned as with the academic licenses. The permission was granted under stricter conditions, all aimed at making sure that the software so released could not be turned into, or used within, proprietary software.

The license was called the **GNU General Public License**, or **GPL**.

In a way, this was a "hacking" of the copyright system. Instead of using the exclusionary rights that copyright automatically grants to the copyright holder to make it a scarce good, thus allowing to charge a price for obtaining it, copyright was used for the reverse effect, to maintain software as Free and unencumbered as possible. Because in English the contrary of "right" is "left", and "left" is a past form of "to leave", someone coined the term "**copyleft**". "Copyleft" is now common parlance in Free Software licensing, a well understood term of the trade.

2.1 Strong copyleft, weak copyleft and non copyleft are the three most relevant categories of Free Software licenses

Software is not built in isolation: reuse of common features, snippets, libraries is a rule. Free Software is no exception, rather the contrary. If software is released under "liberal" licenses, which do not contain copyleft conditions, software can be reused at will within and by software project under any license, and the product of this combination can also be licensed under any license, including a proprietary one. Because this is precisely what a copyleft license does not allow to happen, the concept of **incompatibility** must be introduced, as well as that of "**derivative work**".

In a nutshell, if one takes a bit of software and changes it, the modified software is a "derivative" of the former, meaning that it is a new work under the copyright of the last author, but this last copyright holder must obtain the permission from the former copyright holder not to infringe their rights. In the chain of development of software the original version is found earlier in the flow of changes, and therefore is called "**upstream**", and the further modifications are "**downstream**". In the Free Software world there is no need to have a direct interaction between the upstream and the downstream developer, the related permissions being granted once for all by the public license. The downstream developer only needs to know what license is applied to the software and what conditions are imposed by it. This is called the "**inbound**" license. The inbound license dictates the licensing choice of the downstream developer, or the "**outbound**" license.

The outbound license needs not be the same of the inbound, but at the same time the outbound software distribution must be made in compliance with the inbound license. In other words, one may combine software and distribute such combination only when **the inbound license** is the same or is **compatible with the outbound**. As a rule, the outbound license must be at least as strict as the inbound, or more, otherwise it would not respect the conditions of the latter. This is why a more relaxed license is likely to be compatible with a stricter one, but the reverse is not true, unless special arrangements are made. Using the best naming convention (the first mentioned license is the inbound license, the second is the outbound):

- **no copyleft** can be compatible with both **weak** and **strong copyleft** (and even with proprietary);

48 http://www.gnu.org/philosophy/free-sw.html

- **weak copyleft** is compatible with strong copyleft, and with proprietary and non copyleft only to some extent;
- **strong copyleft** is not compatible with weak copyleft, with no copyleft and with proprietary software outbound licenses. Moreover, different strong copyleft regimes are almost assuredly incompatible with each other.[49]

It is time to define what weak and strong copyleft mean.

The model described at the beginning of this paragraph is a very simplified model, the reality is far more complex. Instead of rewriting each and all of the very basic functions that are almost invariably found in a program (like opening a file, saving a file, displaying an error message, etc.), programmers reuse **libraries** of software, concentrating only on the central part of programming.[50] A library licensed under the GPL would trigger copyleft over the work in which it is used. Developers, including the FSF when it first started releasing the libraries of the GNU C Compiler (gcc) or the GNU C libraries, or glibc, felt that this was too far reaching. They were confronted with the need to produce libraries that could be included in any kind of outbound-licensed software. Therefore a special license for libraries was conceived, the **Library GNU Public License**, or **LGPL**. The LGPL differs from the GPL because the scope of its copyleft is limited to the library itself, not to the entire software which results from the inclusion of the library, or “the larger work”. In other words, instead of covering all the derivative work, the copyleft effect only covers the modifications to the original work. Because of the diminished effect of the copyleft provisions, the LGPL was renamed as “Lesser **GPL**”. This lessened copyleft effect is commonly referred to as **weak copyleft**. The full copyleft is referred to as “**strong copyleft**”.

2.2 How many licenses are there? The proliferation issue

So far a very limited number of licenses have been described: the BSD/MIT in the non copyleft area, the LGPL in the weak copyleft area and the GPL in the strong copyleft area. Other projects have created their own licenses, all of which roughly can be put under the three categories of Free Software, as outlined earlier.

The most popular ones are probably:

- the **Mozilla Public License**, used and “stewarded” by the Mozilla Foundation, the entity behind Firefox and many other projects; it is a *weak copyleft* license.
- the **Apache Public License**, used and stewarded by the Apache Foundation, the entity behind the Apache web server; it is a *non copyleft* license.

These licenses cover the large majority of software out there. But there is a very long tail of other licenses, frequently project-specific and seldom used outside their namesake projects.

The Open Source Initiative (OSI),[51] the organization which collects a list of licenses compatible with the “Open Source Definition”,[52] lists roughly 70 *approved licenses*, but the full list of used licenses is easily one order of magnitude longer.

This phenomenon is often referred to as “**proliferation**”. Proliferation is considered a problem for Free Software due to its three main consequences:

49 See further below for a more detailed discussion of this point.
50 In C programming, the most paradigmatic example, all these libraries are called upon at compiling time and produce different software objects, that are linked together and written in a big executable file (statically linked) or made available alongside the executable file as “dynamically loadable” libraries that are linked when needed by the operating system when they are needed at execution time (dynamically linked). This is a source of complication as to whether using a dynamically linked library creates a derivative of said library, but the discussion of how this works in practice is very sophisticated and exceeds the purposes of this paper, also because it is a source of disagreement between experts in the field.
51 http://opensource.org
52 The Open Source Definition is a list of ten characteristics that must be present in a license to qualify as “open source”. Operatively, this list can safely be considered equivalent to the four freedoms of Free Software.

- **Incompatibility**. With more licenses and more licensing conditions, the odds of software released under legally incompatible licensing conditions increases exponentially. This reduces the chances to reuse software in the commons, by creating reciprocally incompatible commons.
- **Uncertainty**. More licenses means less chances that the scrutiny of the courts validate the working of a particular license. Moreover, while the most used licenses undergo the analysis of legal experts and very detailed discussions amongst them as to what is the scope, validity, meaning of their language, often even before their official release, the seldom used ones are more likely to be totally untested.
- **Confusion, increased transaction costs**. Free Software is supposed to reduce "friction" in sharing and reusing software. The more licenses are relevant to a given project, the more it is difficult to assess whether said project complies with the conditions of each and every license, or to predict the legal outcome of combining different software.

It is apparent from the above discussion that any decision to create a new license, instead of using one of the most popular and thoroughly tested, needs to be taken with due care, because the advantages of creating something that is (in the eye of the drafter) perfected is easily outweighed by the systemic disadvantages of creating *yet a new* license.

2.3 The case of EUPL, relicensing permissions and the exceptions

In this light, I have been very sceptical about the decision of the EU authorities to create a new license, namely, the **EUPL**.[53] On the one hand, I understand the rationale behind it, on the other hand, it suffers from the same disadvantages discussed above in terms of proliferation. Moreover, being a purportedly strong copyleft license, it would be outright (and both ways) incompatible with the most widely used copyleft license, the GNU GPL, and very likely incompatible with many others. This was well understood by the drafters, who decided to use a clever solution to avoid the EUPL being cut off from software development in combination with a large share of the software publicly available.

The EUPL adopts an **express compatibility list.** This overrules some compatibility problems by allowing a larger work containing EUPL code to be "relicensed" under another, otherwise incompatible, license. In other words, the EUPL recedes in front of incompatible licenses, by allowing the incompatible license to become the outbound license of the larger work. In all practical terms, while this can be regarded as a workable solution, the possibility to relicense the software means that the legal conditions actually applied by the downstream developers can be different, and less protective, than those expected. For instance, among the compatible licenses listed in the Annex to version 1.1 of the EUPL there are weak copyleft licenses, like the Eclipse Public License,[54] or non copyleft licenses like the Open Software License.[55]

In a way, the compatibility list makes software licensed under the EUPL multi-licensed. This is not *per se* something to be rejected, as long as the consequences of this being multi-licensed are well understood and predictable. For instance, the GNU GPL has the option to include a clause "or any later version". If this clause is added, then the recipient of the software can, without asking a permission from the upstream copyright holder, relicense the software under a newer version of the same license which has been issued by the official steward of it (so far the Free Software Foundation), as it happened with the publication of version 3.0. The same possibility is given by the Mozilla Public License.[56] The difference between these cases and that of EUPL is that the "any later version" clause only applies to later versions that follow the same strong copyleft regime. When it comes to the EUPL, however, its compatibility list makes it possible for software licensed under a strong copyleft regime (i.e. under the EUPL) to become subjected to weak copyleft, or even to non-copyleft. It is reasonable to assume that if copyright holders choose a copyleft license,

53 EUPL stands for "European Union Public License" http://joinup.ec.europa.eu/software/page/eupl
54 http://www.eclipse.org/legal/epl-v10.html
55 http://opensource.org/licenses/OSL-3.0
56 http://www.mozilla.org/MPL/2.0/ See Section 10

and the more so if they choose a *strong* copyleft one, such as the EUPL is supposed to be, they want to exert control on the legal conditions of their software when it is included into other software. By leaving a backdoor open that allows the software to be relicensed under a license of different nature, the expectations of the copyright holder are somewhat "betrayed".

Another way to introduce compatibility with other software, where compatibility does not exist, is to use **exceptions**. An exception is an added condition that lessens the copyleft effect of (most frequently) strong copyleft licenses, to make them compatible with certain other licenses. One of the most used exceptions is the **linking exception**. As reported above, linking is a widely used form to include libraries into larger works without commingling software, keeping some level of separation at least at source code level. A linking exception would permit to link software licensed under two incompatible licenses and to distribute the resulting larger work under the conditions of the other linked software. Adding a linking exception is obviously permitted only if the decision is taken by the copyright holder(s) of the entirety of the concerned software, and it is akin to adopting a weak copyleft license. Indeed, the GNU LGPL v.3 is now construed as an exception to the GNU GPL v.3 license. The GNU GPL v.3 actually includes a framework to allow adding exceptions, which can be more restrictive than those of the "vanilla[57] GPL" (in certain and limited cases) or more liberal. In case more liberal exceptions are added, it is optional to the downstream recipient to remove them and to distribute their works under the original version of the GPL.[58]

In conclusion, when a project is reported as being licensed under a given license, in order to identify the legal regime that actually applies to it one must consider at least whether:

- The license adopts a compatibility list or the software is dual- or multi-licensed;
- The license and/or the copyright holder allows relicensing under certain circumstances (for instance: it is permitted to relicense the software under a newer version);
- Additional permissions, or exceptions, apply.

3. SAME LICENSE, MANY "BUSINESS MODELS"

Discussing commercial exploitation and monetization (if any) of Free Software and/or how software development is financed is outside the scope of this work. However, some discussion about the social and economic environment where Free Software is made is useful, in the light of the additional legal effects involved in following certain models as opposed to others.

In reality, under the general heading of "Free Software" or "Open Source Software", the matrix of combinations of different characteristics can be very complex. We just mention the most relevant axes in this multi-dimensional space.

3.1 Cathedral against the Bazaar (very few or only one vs. many developers)

A frequent misconception of those who approach Free (or open source) Software for the first time is to believe that it is a way to allow distributed creation of software by many different developers that work in an unorganized, chaotic way, maybe in their garage, to produce something that is half backed, incomplete and inherently difficult to use.

Contrary to that, development of Free Software is not necessarily different from development of software licensed under proprietary conditions. The fact that certain software is Free Software simply allows many more degrees of freedom and choice. Choice creates many different approaches.

57 "Vanilla" is Computer Sciences jargon and means "original as it came from the source, untouched". The Oxford English Dictionary accounts it as "having no special or extra features; ordinary or standard". http://oxforddictionaries.com/definition/english/vanilla?region=uk&q=vanilla
58 See the GNU GPL v.3, Section 7 at http://www.gnu.org/licenses/gpl.html

Perhaps the first discussion of one of these choices has been made by Eric S. Raymond in his work “The Cathedral and the Bazaar. Musings on Linux and Open Source by an Accidental Revolutionary”.[59] In this essay, Raymond analyses the differences between the centralized, rigid and structurally coherent development of GNU, Emacs and GCC by the Free Software Foundation, and the more fast-paced, loosely engineered, and chaotic development of Linux. The two models are indeed at the extreme and both are – judging by the results – quite functional and successful.

One of the differences between the cathedral and the bazaar models is how copyright is dispersed (in the case of the bazaar) or whether it is kept by a relatively close number of holders, because contributors are also a small number. Bringing the concept further, consolidation of copyright into a single entity is also a possibility. This can be done by not allowing others but the original copyright holder to contribute, or through assignment.

Indeed, the Free Software Foundation strongly urges the assignment of all the code committed to the GNU project to itself so that there is ideally only one entity legally entrusted with the management of copyright.[60] This can be advisable (provided that the assignee deserves a high degree of trust) for a number of reasons, one of which being the possibility to take fundamental decisions as to management of copyright, such as adopting a new license or using the ownership of the software as a title to proceed in court against trespassers.

The possibility of taking (or keeping) tight control of the entirety of the software in a project by retaining copyright ownership has been used in different circumstances to achieve models which can be considered hybrid between proprietary and Free Software, as it is the case with dual licensing and “open core” strategies, which will be discussed in the next two chapters.

3.2 Dual licensing

“Dual licensing” (or “multi-licensing”) is a licensing model where software is distributed under more than one license at a time. We have seen how this can happen in order to overcome potential incompatibilities in linking or mixing software. However, using dual licensing as a way to monetize Free Software has peculiar consequences. Dual licensing under a Free Software (most frequently a strong copyleft) license and under a proprietary license is one of the options.

The rationale behind dual licensing is that if software is licensed under a strong copyleft license – therefore a license that does not allow proprietary exploitation of the derivatives – albeit being free (at no cost), its use would bear conditions that could be unacceptable to certain potential downstream developers. These developers, for instance, might want to utilize the copylefted software in their derivative software in a proprietary way. In order to be permitted to do so, they need to be released from the copyleft conditions and therefore they can be in a position to pay a monetary price for this privilege. This can be described as “buying an exception” to copyleft.

The more burdensome and far reaching the conditions are – and the more valuable the software is – the higher will be the incentive for seeking the exception. Since in order to grant an exception the licensor must control the entire copyright of the software, there is a high chance that this will only be possible in the case of a “silos” development model, where everything is made internally by the licensing entity or contracted by it, and such entity therefore holds the entire copyright of the software.

Yet the licensing, even in this concentrated development model, creates two interesting additional side effects compared to an equivalent proprietary development, from the point of view of recipients, as a result of two characteristics of the Free license, namely that:

- software can be “**forked**”

59 http://www.catb.org/~esr/writings/cathedral-bazaar/
60 http://www.gnu.org/licenses/why-assign.html

- software can be inspected, modified and adapted to the customer's needs without asking permission to the licensor.

The fact that software can be forked means that downstream recipients could decide that they can take over the development of the same solution and continue its development and distribution only relying on the inbound license. This alone creates a *game* where there is a certain assurance that development will be continued even against the will of the copyright holder, or in case this latter is unable or uninterested in bringing it forward. This removes some of the uncertainties that always exist around proprietary products, including that they can become “orphans” or that they can follow undesirable development paths. Indeed some of the largest dual licensed software projects have been forked over concerns as to the stewardship of their copyright holder: Libreoffice has been forked from Openoffice.org (formerly owned by Sun Micrososystems, which was taken over by Oracle Corp; presently the stewardship is with the Apache Foundation) and MariaDB was forked from MySQL (the most successful Free Software database application and one of the most successful databases at all) by Monty Widenius, its original creator. Forking is indeed one of the most important (and feared, by the leaders of the projects) safety measure against misconduct or simply a rather drastic way to resolve disagreements.

The second aspect, the freedom to modify the software and adapt it to the user's needs, is also interesting. Besides allowing anybody who receives the software to perform an independent inspection, this freedom encourages sufficiently literate users to fix bugs by providing patches, or even producing drivers and other tools. Because these modifications can be “broken” by changes in subsequent versions, it is highly efficient that these modifications are given upstream to the copyright holder, so that they are included in the development and maintained over new versions.

This can only be done in two ways: by assigning the copyright of the modification, or by releasing the patch/additional software under ultra-liberal conditions, so that they can be included in both the Free and the proprietary version of the mainstream application without depending on the conditions imposed by the contributor.

Although it is not necessarily so, it does not seem that more important contributions than patches and bug fixes are ever given to dual licensed projects. In fact, it is very hard to find dual licensed projects that receive substantial contributions, so that the development of those projects hardly leaves the shoulder of the main copyright holder.

3.3 Open Core (Free core platform or infrastructure, proprietary outer layers)

On the opposite side there seems to be something that is regarded as increasingly popular, which is open core. If in the dual licensing model the bulk of the software is kept and maintained by one single entity, and something comes from third parties only at the edges, in open core the fundamental technology is released freely, and the proprietary part happens at the edges. In other words, there is an attempt to build a shared infrastructure (“**the core”**) with **differentiation** and added value only occurring on the outer layers, or the actual implementations based on the common infrastructure, akin to standardization.

There are examples where a common shared infrastructure is made by many independent companies and/or communities of developers under very liberal, non copyleft conditions. PostgreSQL (an increasingly popular database application) is an example: it is licensed under the BSD license, which allows any company to sell an entirely proprietary application based on it. Sometimes the common infrastructure is conversely kept under very strong copyleft conditions, to avoid that any forks are taken away from common development, and only the “outer layer” is distributed as proprietary add-ons.

Linux can roughly fall under this category, as it is used both in the GNU/Linux incarnation (for the PC systems, in desktops and servers) and as the foundation of proprietary solutions built on the top of it; the most widely known is perhaps Google's Android, mainly used to run smartphones and tablets. Android is interesting also because it creates a middle layer of software released under very liberal condition (similar to the Apache license) that can be taken by all vendors to make their own tight-controlled versions, bundled with their own hardware.

3.4 Community vs. company controlled development

The above examples show how proprietary exploitation and one-company development are not necessarily tied together. On the one extreme, FSF has strong control over GNU, but absolutely no proprietary exploitation. On the other extreme there is large proprietary exploitation through selling of copies of software with a project run under very liberal and non centralized ways in PostgreSQL. On yet another axis, we have bazaar-like development of software under strong copyleft conditions (Linux is one example, KDE is another), and one-company-controlled development of software under very liberal conditions (in the case of Android).

Then again, it is possible to have consolidation of copyright into a single entity while retaining a bazaar-like development model. KDE is such an example,[61] where developers are at least invited to provide an assignment to KDE e.V., through the "**Fiduciary Licensing Agreement**" (or FLA)[62] which is a legal instrument developed by the Free Software Foundation Europe, and whose aim is to centralize copyright title while preserving the distributed nature of copyright interest – and the nature of Free Software of the code whose copyright is assigned. In a FLA the individual assigners are the "beneficiaries", whereas the assignee, who receives a title in the software, acts as a trustee of the beneficiaries.

4. ARE FREE SOFTWARE LICENSES VALID? WHAT MAKES THEM ENFORCEABLE?

The question of the validity and enforceability of Free Software licenses is not contested anymore. The issue has been raised several times and addressed in multiple courts which ruled in favour of enforceability of free software licenses. As a license is frequently considered as an agreement, contractual laws of various countries can work against it, on many grounds, like formal deficiencies, lack of consideration, consumer protection, language preservation laws.

But are licenses "contracts"?

Let us start from what the most famous family of Free Software licenses say. The GNU licenses have a very telling section commonly referred to as "not a contract". The drafters had the issue of different contract laws very clear, and decided not to rely on any particular contract law in order to preserve the validity and enforceability of the license. Instead, they relied on the uniform law provided by copyright treaties, mainly the Berne Convention.[63] Those treaties grant a minimum set of protection to copyright holders, among which are the rights to exclude others from use, copy, distribution, modification of software. Were the license absent or invalid, no right would be transferred. If recipients of the software were to contest the validity of the license, they would automatically plead the fact that they are infringers of the copyright of all copyright holders.

Most of the licenses provide a set of conditions, rather than obligations. This makes any contractual qualification of the license redundant,[64] as the license can very well work as a

61 KDE stands for K Desktop Environment. It is one of the most popular desktop environments for GNU/Linux. A Desktop Environment is the part of the operating system that provides its Graphical User Interface (GUI), including the windows within which applications are displayed, their decorations, the "desktop" where icons representing various objects reside, and a number of applications, tools, snippets, utilities, configurations etc. that assist the user in interacting with the computer's operating system. KDE also provide an integrated suite of applications that are particularly designed to be executed in the KDE environment, such as personal and office productivity applications, an email client, a web browsers and many other utilities. http://kde.org
62 http://ev.kde.org/rules/fla.php.
63 See, for a clear an concise explanation, Eben Moglen, *Free Software Matters: Enforcing the GPL, I* http://moglen.law.columbia.edu/publications/lu-12.html
64 As I already maintained in one of the first articles in Italy on that matter: Carlo Piana, *Licenze pubbliche di software e contratto*, in I contratti, n. 7/2006, IPSOA; available at http://www.piana.eu/repository/720_727.pdf

"bare copyright license". Licenses do not require licensees to do anything. They provide a set of conditions: only fulfilling all of them gives the licensee the right to use, modify, copy, distribute, etc., the software.

In the **United States** a landmark case from the Court of Appeal for the Federal Circuit gave the same interpretation, allowing injunctive relief in a case of violation of a Free Software license, in *Jacobsen v. Katzer*.[65] In **Europe** German courts followed the same line of arguments in several cases raised by gpl-violations.org, despite following the contractual line in the first place.[66]

If anything, the stricter the license is, the more evident is the intention of the copyright holder to retain control of the software, as opposed to letting it fall in a "public domain". Therefore, copyleft conditions strengthen the license and provide more ground for its enforcement.

4.1 In particular: the liability regime and exclusion

Typically, FOSS licenses contain very strong **disclaimer** clauses, which discharge the author from all liability.[67] The reason for this is that FOSS is often made available without any monetary compensation of any sort, as a result of which the author generates insufficient income to pay for liability insurances and legal costs.[68]

Should the disclaimer be ineffective, could a software developer be liable for damages caused by his or her software, in the light of the fact that his or her software is released for free (under the Free Software license)? Apart from the cases of gross negligence and intentional acts, or a liability in tort (such as releasing malicious software), the answer seems to be negative in most cases. It is quite hard to construe a contractual liability only based on the FOSS licensing. In a typical software license there is no obligation to deliver, just conditions for use. Should a downstream recipient wish to integrate the software in a larger product for a particular purpose, and the software be unfit to said purpose, it would be upon the integrator – who is permitted to make all the modifications needed, including the adaptations and quality assurance activities – to make sure that the combination works.

There is a considerable difference between this case and a proprietary software license. In proprietary software licensing, consideration is exchanged against the delivery of software or even just against permission to use said software, which is to be qualified a *sale*.[69] As a sale, it bears certain statutory warranties, including that the product is free from defects that reduce its intended use. The same cannot apply to FOSS, which is not "sold", but just offered to public use. If there is a separate agreement, such as a software development agreement, the relationship between the client and the developer – in particular the liability for defective software – is governed by this specific contract and not by the license.

The above conclusion is true when the **contractual** relationship between the upstream and the downstream parties in the transaction is the license alone. It might well be the case that separate agreements (like a software adaptation contract or a maintenance agreement, or even a sale of a device containing software to an end user) are in place, in which cases contractual liability rules would apply to that part of the relationship.

65 Lawrence Rosen, *Bad facts make good law: the Jacobsen case and Open Source*, IFOSS L. Rev., 1(1), pp 27 – 32. Available at http://www.ifosslr.org/ifosslr/article/view/5

66 Landsgericht Frankfurt, Case 224/06 www.jbb.de/urteil_lg_frankfurt_gpl.pdf An English translation is available at http://www.jbb.de/judgment_dc_frankfurt_gpl.pdf

67 See e.g., the BSD license (http://www.opensource.org/licenses/bsd-license):

"THIS SOFTWARE IS PROVIDED BY <copyright holder> "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 <copyright holder> 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."

68 B. PERENS, "The Open Source Definition", Open Sources: Voices from the Open Source Revolution, http://perens.com/OSD.html

69 Cfr. European Court of Justice, case C-128/11 UsedSoft v. Oracle, par. 44-72

Liability for **lack of title** is also a possibility. The releasing of software as Free Software by an upstream provider could have been relied upon by a downstream recipient. If there is a hole in this chain of title (e.g. in case one developer has taken software without complying with the licensing conditions of it), this could result in the lower end of the chain being damaged, e.g. because of resulting litigation. Can this distributor of software demand to be indemnified by its upstream software provider who has "obfuscated" the real status of the copyright title of that particular piece of code? Such indemnification is hard to construe because there is no contractual link between the party requesting indemnification and its upstream provider. What remains, in the absence of express warranties and representation, is non-contractual liability. Certainly the licenses have no warranties and representation, rather the contrary. Due diligence, in this case, seems to be the only protection one has. In fact, there is a growing business for consulting companies which also offer various kinds of automated source code (and sometimes also object code) checking tools and who maintain a large database of referenced code. SPDX statements[70] or other licensing meta-information can be used to this effect.

Finally, liability could be claimed on **tort**. It is reasonable to believe that a principle of "*caveat emptor*" (for want of a better wording describing a principle that would place the risks on the recipient of software) in a Free Software distribution could also be held to apply. Therefore, only the final user who receives software as a part of a device or of a software distribution could be in a position to claim damages should the software be defective, and only vis-à-vis the party who has compiled the code. The chain of liability would end pretty soon, as the code is "inspectionable" – unlike what happens in a proprietary setting. This chain would be interrupted as soon as a provider has had a chance to inspect the code to verify its malfunctioning.

The only generally applicable limit to liability disclaimers contained in a Free Software license seems to be found, in Europe, at least, in consumer protection law (which puts strong limits to the effectiveness of liability disclaimer in consumer agreements)[71], then again, this should be considered in the light of the particular licensing of the software.

5. INTERACTION WITH OTHER "IP" RIGHTS

Free Software licensing heavily relies on copyright, but it would be plainly wrong to say that licenses are just copyright licenses. The rights that are received by the users/recipients go well beyond copyright. At the same time, other rightsholders – sometimes unrelated to the actual development of the software – can claim rights that could interfere with the free working of the licenses and contradict the freedoms that they can ensure.

5.1 Patents

Art. 52.2 (c) and 52.3 of the European Patent Convention[72] seem to exclude software from the subject matters that can be patented. However, the constant practice of the European Patent Office and of some of the Member States' patenting offices has been in stark contrast with the letter of the Convention, so that there exist patents that read on pure

70 "The Software Package Data Exchange® (SPDX®) specification is a standard format for communicating the components, licenses and copyrights associated with a software package". See https://spdx.org for more information.

71 For instance, implementation of the Directive 93/13/EEC of 5 April 1993 on unfair terms in consumer contracts http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31993L0013:EN:HTML.

72 "*Article 52 - Patentable inventions.*

(1) European patents shall be granted for any inventions, in all fields of technology, provided that they are new, involve an inventive step and are susceptible of industrial application.

(2) The following in particular shall not be regarded as inventions within the meaning of paragraph 1:

[...]

(c) schemes, rules and methods for performing mental acts, playing games or doing business, and programs for computers;

[...]

(3) Paragraph 2 shall exclude the patentability of the subject-matter or activities referred to therein only to the extent to which a European patent application or European patent relates to such subject-matter or activities as such."

software, namely software distributed as such and made to work in general purposes PC, not being embedded or as a component of any special purpose machine or apparatus.

This is not the place for a full discussion of how this practice seems to be illegal and directly against the Convention – or simply nonsensical and detrimental. Let us just assume that the practice is effective and patents can be enforced against pure software distribution. Exclusionary rights conferred by patents can be used by patent holders to stop or impair distribution of software, through legal means or other kind of impending threats.[73]

This is also not the place to discuss how – as matter of fact, and as it is a common way of saying – the issued patents on software are more likely to “patent the problem” itself rather than a solution to a problem, are overreaching and poorly described. But it is common experience that it is almost impossible to find which patent is relevant to which piece of software, because of how patents are issued and the language which is used and their breadth of scope, as well as – or perhaps mainly – the nature of software.

5.1.1 Patents are enemies of Free Software

The very existence of software patents is negative and **in conflict with Free Software**. Under copyright, as it is commonly interpreted, the authors of an original work are almost sure that the copyright is only their own. If they reuse software coming from other sources, they can safely rely on the inbound license and if the license is a Free Software one and compatible with the outbound license of their software, they can assume they are not trespassing on anybody else's copyright.

Under patents, neither of these two tests (originality, compatibility) is likely to be sufficient or passed.

- Copyright **covers the implementation** of an idea, so if two independent authors come to the same clever idea, they will very likely have different code, unless one copies from the other. Patents **cover the idea** itself, so if one of the authors above is granted patent protection, the other cannot use their own code, unless a separate patent license is obtained.
- Most of patent licensing schemes have conditions that are directly in contrast with the very working of Free Software. One for all, the **running royalties** contradict the freedom to make copies and to distribute them. The very obligation to **report sales** requires that distribution is controlled, which is *per se* contrary to the basic Freedoms. Only very broad “patent promises”, frequently construed as “covenant not to sue” can be considered compatible schemes.

Frequent objections to these arguments use some of the following arguments. The first is that the other software developer can “**invent around**”, so that their software is made in a different way that does not infringe the patent. This is true, of course, in theory, but the objection does not take into consideration that patents are most often too broad and, because they mainly “protect the problem rather than the solution”, any other solution to the same problem would also be an infringement. Moreover, inventing around is precisely impossible when the patented invention is enshrined in **standards**. If this is the case, it is almost assuredly impossible to comply with the standard *and* avoid the patent. The common parlance is that the patents are “**necessarily infringed**”. Only Royalty Free standards (or more precisely, standards whose implementation does not require per-copy royalties, or “running royalties”, provided that the license does not impose other incompatible conditions) are compatible with Free Software and thus can truly be considered “Open Standards”. Running royalty-bearing **RAND**[74] and **FRAND**[75] conditions (at least as they are interpreted by most of patent holders), frequently associated with the “open standard” term, are impossible to be made compatible with Free Software.

73 Such as a practice known as “patent FUD”, from FUD = Fear, Uncertainty, Doubt, a marketing technique which is well described in http://en.wikipedia.org/wiki/Fear,_uncertainty_and_doubt. This practice basically consists in forcing competitors and/or users of competing technology to enter into a licensing agreement or to stop using the competing technology by implying that contesting even unspecified threats of litigation will be unreasonable and anti-economic.

74 Reasonable And Non Discriminatory

75 Fair Reasonable And Non Discriminatory

The most frequent rebuttal to the above is perhaps that the patented part can be embedded into a proprietary, patent-license-compliant part, which can be used to implement the function, while the rest remains Free Software, via an intermediate layer of software that keeps the two bits at arm's length. This process is referred to as “**shimming**”. Another way to achieve similar results is using “**plugins**” into a modular system. While this is indeed possible and a used workaround, it is very likely to create an unnecessary additional complexity on the one hand, and on the other hand it only allows making Free Software that *consumes* the result of implementations of the standard, but that does not *implement* the standard. It is Free Software that *makes use* of proprietary software. The consequence is that, the more patent-ridden standards come into existence, the more the breathing space for Free Software shrinks.

Proprietary software “used” by Free Software cannot even be distributed in the same way as Free Software, or under an overall Free Software license, but needs to be acquired separately by each and any of the recipients of Free Software, which is therefore heavily crippled or deprived of much of its beneficial effects.

5.1.2 Patent provisions in Free Software licenses

As soon as the issue of software patent became a real nuisance, new licenses started to tackle the problem with certain provisions to ensure that what is given with the “copyright license” is not taken away with the use of the patent.

The most common provisions are:

- patent retaliation clauses (or termination clauses);
- implied or express patent licensing clauses.

A patent retaliation clause simply terminates the Free Software license in case the licensee uses the patent to claim exclusionary rights against the covered software. So if A receives software X from B under a license that contains the termination clause, and distributes it (or a modified version thereof) and at the same time A requires B or any other recipient to receive a separate license on the patents A claims on the same software X, B, as copyright holder of X, can terminate the license, so that A becomes an infringer on B's copyright.

The implied or express license works on a different level. If A receives software X from B and distributes it (or a modified version thereof), all recipients from A will receive a license on the patents controlled by A that are relevant to X, so that if A were to claim patent rights over X, B and all others could claim that they are already licensees for them.

The express license can come under two species:

- the license only covers the additions made by the patent holder (this is the case of MPL), so that modifications made by others do not trigger a license on patents that the contributor owns and that read on parts of the software that the same contributor has not modified;
- the license covers all patents that read on distributed code, regardless whether contributed by the patent holder or just received and distributed without modifications (this is the case of GPL v.3)

In no known cases is the express patent license triggered by modifications made by third parties where said modifications are not distributed by the patent holder.

5.2 Trademarks

Trademarks are used to identify a product or service, and/or their provenance, with a name, a symbol or other signs, so that their identity is readily established and recognized by the public. A trademark, however, is not necessarily associated to a Free Software product. If the software is maintained by a group of developers or a foundation, the name under which it is distributed can be claimed as being a *de facto* trademark, or a common law trademark under certain jurisdictions. This is important when forks occur, and a clash between the original group and the forking group (or other entity that can claim the use of the sign) can occur. In other cases, the entity behind the software development registers and uses a trademark to identify its software products, under more or less strict trademark policy, sometimes not even expressly fleshed out. Since the same Free Software product is very likely taken and modified, even substantially, by third parties, this can lead to confusion and can even have consequences on the validity of the registered trademark.

For this reason, some entities have implemented very strict policies. For instance, the Mozilla Foundation requires that the name “Firefox” can be kept to identify only those versions of the software that have been compiled and packeted by it, regardless of who distributes them. Therefore, if someone takes the source code of Firefox and recompiles it, even without modifying the source code, they cannot call the resulting product “Firefox”, but they must use a different name. In fact, since the Debian Linux distribution repackages it as a part of its quality assurance process, the version distributed with Debian is called “Iceweasel”.

Another notable example is Red Hat. Red Hat manages one of the most widely adopted Linux distributions. Because of its stability and quality, it is a certified target distribution for enterprise-level software applications. Since it is entirely Free Software, there are perfectly legitimate “clone” distributions, distributions that use the same codebase as Red Hat with small additions and changes (CentOS and Oracle's Unbreakable Linux are the most famous examples) so that they are almost entirely compatible with Red Hat and can easily become certified, in case. Contrary, or in addition, to the use case of Firefox, where restrictions on the use of the trademark are mainly due to quality assurance and control over the trademark, Red Hat uses the trademark as a business tool for selling services, which can only be used by those who deploy the Red Hat distribution.

Tight control of trademark is not incompatible with Free Software, rather the opposite. Since trademark law has “fair use” concepts, use of the originator's trademark is generally permitted to indicate provenance of the code, if other requirements of fair use of trademark are complied with. This includes the permission to reference the trademarked software to indicate that the other software is a derivative of it, a reimplementation, a drop-in replacement, a compatible alternative, etc. As we have seen with a notable example, trademark is one way to monetize software development and quality assurance in a 100% Free Software distribution model.

5.3 Database rights

Database rights are *sui generis* rights (rights of their own kind) granted to a collector of a set of data when relevant investment has been put into creating the database. They do not relate to the copyright of the content of the database, or to the peculiar way the database is construed. Finally, they do not relate to the database software that can host a database, which is a software application like all others.

There is no particular interaction between database rights and software. It might happen that a software distribution contains databases, but it is very hard to find relevant cases in the licensing of Free Software, especially if the database – even if it is required – can be replaced with an empty or meaningless one (so called “dummy” database). An example can be content management software which comes with a database of configurations. In this case it is not a database as such, but just a set of configurations that have no meaningful purpose outside the application itself. In any event, as mentioned earlier, since this is an integrated part of the application, the database rights can safely be considered under the same license as the whole application, unless specific provisions in the licensing language carve the database out of the license grant.

6. THE CLOUD

For the purposes of this discussion, “Cloud” refers mainly to Software as a Service (SaaS), which is the farthest end of the different kinds of cloud computing. SaaS subverts the traditional way of distributing software. Leaving aside the technical implications, for the sake of this discussion SaaS delivers the same useful effects of software not by distributing code – be it in object, source code or other forms – but by providing remote access to interfaces and services of software executed on servers that are run by third parties, or service providers.

If the relationship between a software maker and a software user is a license (express or implied) under copyright, the relationship between the software service provider and the user is a service contract and the performance of this contract is measured in service levels and availability. There is no exchange of code and there is no direct relationship, or rights, of the end user on the code. There are also no statutory protections granted by software legislation (such as the right to make backup copies or to study how the software works or to decompile it), to date.

Free software is widely used to power many cloud services,[76] yet customers of these services cannot benefit from the Freedom that they would benefit from if they used software directly. They certainly have no rights to demand to receive the source code, which is the foundation of most of the Freedoms of Free Software, because SaaS is not a distribution of software, while copyleft is triggered by distributing modified copies of the software.

6.1 A different license, The Affero GPL, or the AGPL

To close the gap, which at the time was referred to as “the ASP loophole” (“the Cloud” was not yet a buzzword), Bradley Kuhn, a developer and Free Software activist, devised an addition to the GPL, that was christened “Affero clause”, in cooperation with at-the-time counsel to FSF Eben Moglen.[77]

In the Affero clause, the copyleft effect is not triggered by distribution, but relies on the right to control modifications. Therefore, anytime software is modified, even if the code is not distributed, but it is consumed through a network interface, there must be a convenient facility where the corresponding source code is included.

The Affero GPL, initially being just an “unauthorized” variant of the GPL v2, is now an officially recognized license of the Free Software Foundation and goes under the name of GNU AGPL v.3. AGPL v.3 is made compatible with the GPL v.3 through an express compatibility clause.

76 Google is a notable example: for references see http://en.wikipedia.org/wiki/Google_platform. But Twitter, Amazon, Facebook, Rackspace etcetera all are based on FOSS technologies, which in several cases they make and distribute as Free Software.

77 References available at http://en.wikipedia.org/wiki/Affero_General_Public_License

REFERENCES

Fondamental reading:

- Daffara, C., Gonzalez-Barahona, C., Jesus, M. (Ed.), (2000), *Free Software/Open Source: Information Society Opportunities for EU*, European Working group on Libre Software, http://eu.conecta.it/paper/paper.html.
- Lindberg V. (2008), *Intellectual Property and Open Source: A Practical Guide to Protecting Code,* Oreilly, California, USA Fontana R., Kuhn B.M, Moglen E., Norwood M., Ravicher D.B., Sandler K, Vasile J., Williamson A. (2008), A Legal Issues Primer for Open Source and Free Software Projects, SFLC, New York, https://www.softwarefreedom.org/resources/2008/foss-primer.pdf
- Meeker H.J. (2008), *The Open Source Alternative*, Wiley, New Jersey, USA.
- Rosen L. (2005), *Open Source Licensing*, Prentice Hall, New Jersey, USA.
- Van den Brande Y., Coughlan S., Jaeger T. (2011), *The International Free and Open Source Software Law Book*, Open Source Press, Munich.

Other references:

- Free Software Foundation, *What is Free Software – The Free Software Definition*, http://www.gnu.org/philosophy/free-sw.html.
- Eben Moglen, *Free Software Matters: Enforcing the GPL, I*, http://moglen.law.columbia.edu/publications/lu-12.html.
- B. Perens, "The Open Source Definition", Open Sources: Voices from the Open Source Revolution, http://perens.com/OSD.html, http://opensource.org/docs/osd
- Carlo Piana, *Licenze pubbliche di software e contratto*, in I contratti, n. 7/2006, IPSOA; available at http://www.piana.eu/repository/720_727.pdf.
- Eric S. Raymond, *The Cathedral and the Bazaar. Musings on Linux and Open Source by an Accidental Revolutionary*, http://www.catb.org/~esr/writings/cathedral-bazaar/.
- Lawrence Rosen, *Bad facts make good law: the Jacobsen case and Open Source*, IFOSS L. Rev., 1(1), pp 27 – 32. Available at http://www.ifosslr.org/ifosslr/article/view/5.
- Sam Williams (2002), *Free as in Freedom. Richard Stallman's Crusade for Free Software,* O'Reilly, California, USA.

Carlo Piana is an Italian Information Technology lawyer and a digital freedoms activist. Based in Milano, Italy, and a member of the local Bar, serves as the external General Counsel for the Free Software Foundation Europe (http://fsfe.org). He is member of the Editorial Committee of the International Free and Open Source Software Law review (http://ifosslr.org), and member of the Board of EuroITcounsel -- a network of European lawyers specializing in IT law. For more information: http://www.arraylaw.eu

Legal aspects of free and open source software in procurement: guidelines developed at the EU level

Rishab Ghosh, UNU-MERIT

ABSTRACT

This briefing paper examines issues around the public procurement of software distributed under free/open source software licenses. It looks at public procurement regulations, the state of current software procurement in Europe, and provides guidelines for best practices for public procurement of open source software. It draws on previous publications of the author, including the "Guideline on public procurement of Open Source Software" published by the European Commission

CONTENT

EXECUTIVE SUMMARY

European governments are increasingly considering the use of Open Source Software (also known as Free Software or Libre Software, or FLOSS[78]) as a means of reducing costs and dependency on vendors, while increasing transparency and sustainability. A number of debates have taken place on the costs and benefits of open source software, and much discussion and interest has been expressed from the perspective of information technologists.

This briefing paper is drawn from previous publications of the author, including the "Guideline on public procurement of Open Source Software" [79] published by the European Commission as part of the Open Source Observatory and Repository (OSOR). Here, open source software is considered not as a matter of technology, but as a matter of public procurement.

This briefing paper explains why it may be useful for public agencies to acquire open source software, and more importantly, how they can do so within the current procurement regulations, once a decision is made.

78 Free Software and Open Source Software, which may be used interchangeably when referring to software, are defined by the Free Software Foundation and the Open Source Initiative. They refer to software that is available under terms that allow users to use the software for any purpose; to study the software source code; to modify the software; and to distribute the software and modifications. See www.fsf.org and www.opensource.org

79 Ghosh, R.A., Glott, R., Schmitz, P., Boujraf, A. (2010). Guideline on public procurement of Open Source Software. Brussels: European Commission

This briefing paper shows how open source software can even be downloaded free of charge without a call for tenders, and provides criteria that can be included in tenders to ensure good practice procurement of software.

Public procurement of software has long been far from a "level playing field", and widespread preferences in public tenders for specific, named, proprietary software and their vendors is one justification of why this paper is useful.

This briefing paper examines the following areas:

- Software public procurement landscape
- Software public procurement needs & principles
- How open source can be a functional requirement
- Downloading software vs procurement through tenders
- How open source functional requirements translate to tender requirements

This briefing paper is about procurement of software, but it should be noted that one of the properties of open source is that it promotes collaboration and participation, rather than just consumption through public procurement. The EU's own Open Source Observatory (now available on the JoinUp portal at joinup.ec.europa.eu) provides a platform for open source software collaboration among public agencies in Europe.

1 INTRODUCTION

1.1 Software public procurement landscape

At the European level, there are no binding policies on open source software procurement, although there are some at the level of Member States or regions. The guidelines described in this briefing paper are applicable in any context within EU Member states, regardless of the existence of any policy, following European procurement regulations alone, with no need for any specific open source policies.

Anti-discrimination (“equal treatment”) is a general principle of procurement regulations, but this refers to equal treatment of potential economic operators: vendors of solutions meeting tender requirements. It does not refer to technical standards, or licensing regulations for software – i.e. it is possible to specify particular licensing requirements (such as open source licensing) when that is required and justified for the purposes of a tender.

Previous studies[80] provide evidence which suggests that in public procurement of ICT, practices that “*do not constitute an equal treatment of all economic operators*”[81] are apparent in tenders on more than an “exceptional basis”, as required by EU law[82]. In particular, these studies have pointed out the frequency of mentioning the names of specific companies and their products, or to require compatibility with previously purchased proprietary ICT products.

In one study, a keyword search for tenders on TED, the EU’s public procurement tool, showed that 149 tenders included brand names.[83] Another similar sampling on a larger

80 Cf. OpenForum Europe (2008). OFE Monitoring Report: Discrimination in Public Procurement Procedures for Computer Software in the Member States. Brussels: OFE.; Ghosh, R.A. (2005). An Economic Basis for Open Standards. FLOSSPOLS project. Brussels: European Commission.; Ghosh, R.A., Glott, R., Schmitz, P., Boujraf, A. (2008). OSOR Guidelines public procurement and Open source Software. Public Draft Version. Brussels: European Communities.

81 European Union (2011). Guidelines for public procurement of ICT goods and services - SMART 2011/0044. Tender Specifications. Brussels: European Union.

82 Reference to specific products or sources is allowed on "*an exceptional basis, where a sufficiently precise and intelligible description of the subject-matter of the contract [in functional terms or with reference to European standards] is not possible*", Directive 2004/18/EC, Article 23(8).

83 Ghosh, R.A. (2005). An Economic Basis for Open Standards. FLOSSPOLS project. Brussels: European Commission.

scale, published on OSOR[84] "*showed that 567 of 3615 software tenders (16%) between 4 January 2006 and 30 August 2008 contained one or more of the top 10 software brands*"[85]. In addition, the study cited showed that the top company in the list was clearly dominant, with a 36.1% share of those tenders that did specify brand names (the second company was mentioned in 20.2% of the tenders)[86].

The European Commission, in a call for tenders for a study, quoted these figures to note that such practice "doesn't constitute an equal treatment of all economic operators who could potentially deliver the product or service that is asked for. Therefore it is allowed only exceptionally. 16% of the cases seem to imply more than an exceptional use of brand names"[87]. Such procurement practice can stem from several reasons. As previous research on public procurement of software in European national administrations has shown, it can be based on a conscious decision to favour compatibility of newly purchased software with proprietary systems that were previously in use[88]. It can also be based on lack of awareness and knowledge of how to adhere to practices in public procurement of ICT that are in line with the European regulatory framework. [89]

1.2 Public sector needs: transparency, sustainability, cost-effectiveness

Public sector consumers of software have an obligation to support interoperability, transparency and flexibility, as well as economical use of public funds. When it comes to public procurement, the principles applied to the public sector require them to support (and certainly not to harm) competition through their procurement practices.

They are also obliged to avoid explicitly harming competition in the market of private consumers. Thus, public agencies should not require citizens to purchase or use systems from specific vendors in order to access public services, as this is equivalent to granting such vendors a state-sanctioned monopoly.

They are also obliged to ensure the best cost-to-service ratio over the long term.

These principles are not only the basis for policy documents such as the *European Interoperability Framework;* they are also implied by the legislative framework governing public procurement, such as Directive 2004/18/EC on public supply contracts and Directive 2004/17/EC on utilities, and Directive 98/34/EC on the provision of information in the field of technical standards and regulations[90].

2 PROCUREMENT PRINCIPLES

2.1 Open Standards

Good practice eGovernment services should provide access based on open standards – defined below - see section 0– as standards that are implementable by all potential providers of equivalent technologies, including open source software. In particular, government should never require citizens to purchase or use systems from specific vendors in order to access public services: as described above, such a requirement would be equivalent to granting those vendors a state-sanctioned monopoly.

84 Ghosh, R.A., Glott, R., Schmitz, P., Boujraf, A. (2008). OSOR Guidelines public procurement and Open source Software. Public Draft Version. Brussels: European Communities.

85 European Union (2011). Guidelines for public procurement of ICT goods and services - SMART 2011/0044. Tender Specifications. Brussels: European Union.

86 *Supra note* 84

87 *Supra note* 81

88 *See supra note* 83. Naming brands or vendors is not the only way to favour specific vendors; the European Commission previously noted that hardware procurement discriminated in favour of Intel by mentioning specific processor clock rates without explicitly naming "Intel". EC Press release IP/04/1210, October 13, 2004

89 Ghosh, R.A., Glott, R., Schmitz, P., Boujraf, A. (2010). Guideline on public procurement of Open Source Software. Brussels: European Commission.

90 These were specifically referred to by the EC in its announcement regarding the investigation on public procurement of computers, concerning tenders specifying "Intel or equivalent". EC Press release IP/04/1210, October 13, 2004.

The precise definition of the term 'open standards' is less important than a clear expression of the reasons why open standards are desired in the first place. These reasons should form part of the requirements for any procurement.

For procurement of software in general, it is good practice for public authorities to implement software based on open standards – as defined by their economic effect of fostering a fully competitive market[91]. Supporting technologies without considering their degree of openness and their ability to foster a fully competitive market is harmful to competition and net social and economic welfare. It is thus expensive, by definition, over the long term. While software based on open standards may not always be available, public agencies should encourage its development, and indicate their preference for open standards to vendors though preferential procurement of software based on open standards wherever it is available. Similarly, public agencies should use open standards wherever supported by the software they implement, in preference to any other technologies supported by such software.

The main advantage of open standards is the capacity to be interoperable with other software systems. By definition, a software application based on open standards is fully interoperable with any other application using the same standards, and it is possible for any other application to use the same standard. By consistently requiring and using open standards, software buyers try to achieve “vendor-independence”, which is to retain the ability to change software products or producers in future without loss of data or significant loss of functionality. This is achieved because one vendor’s software, if it is based on open standards, is fully compatible with other software available from other vendors; therefore, the customer does not get locked-in to that vendor simply because of the standards used. Data created with that software is still fully usable with software from another vendor.

However, this goal is often incompatible with implicit or explicit criteria for software purchasing, in particular those requiring that new software should be compatible with previously purchased software. Buyers who use the latter criterion rather than a general requirement for open standards or vendor-independent interoperability in effect remain locked in to their previously purchased software. Thus, even if they see the benefits of open standards and believe in interoperability, buyers whose preference for new software is based on compatibility with previously installed software are not, in practice, supporting or benefiting from interoperability.

2.2 Open source software

Open source software, Free Software, or *libre* software, also called FLOSS, is software that a user can:

- use for any purpose
- study, by examining the source code
- modify and improve
- distribute, with or without modifications

This basic definition of FLOSS is equivalent to the *Four Freedoms* of the Free Software Foundation (FSF, which officially defines "free software") and the *Open Source Definition* maintained by the Open Source Initiative (OSI).

Open source software is copyrighted by its authors, and is made available under copyright licences that provide the freedoms required by the above definition.

Most major free software or open source licences have gone through a formal process of approval by the Open Source Initiative, and are listed on the OSI website; these licences are OSI certified and authorised to use the "Open Source Initiative Approved License" mark. Of course, licences that meet the terms of the Open Source Definition but have not been formally processed by the OSI (and thus not listed on their website) are also open source licences.

91 Ghosh, R. A. 2005. "An Economic Basis for Open Standards". FLOSSPOLS project, European Commission.

2.3 Procurement principles and sustainability

Open standards and open source software, as separately outlined above, are both relevant to the procurement principles described previously. When based on open standards, open source software supports the sustainability of government ICT processes and systems through:

- **transparency and security**: open source software is available along with its source code which can be studied and modified. This can ensure the security of the software, as its processes are examined in detail under widespread scrutiny and improved. Open source software also allows appropriate stakeholders to understand and monitor the functioning of government processes that are implemented in software - for instance, to ensure that voting systems are calculating results correctly.
- **interoperability**: whether implemented in open source or proprietary software, open standards ensure interoperability, the ability of systems from different vendors to function fully with each other without technical or legal obstacles. Open source software, in particular, provides additional support for interoperability, as its processes can be studied and adapted to work with other systems.
- **independence**: transparency and interoperability allow current and future vendors to work with, adapt and maintain the software, eliminating the dependence of purchasers or third party support and service providers on the vendors of the original version of the software.
- **flexibility**: open source software allows systems to be adapted and extended as user needs evolve. It does this without requiring that the user go back to the original vendor - new suppliers can be selected on a competitive basis.

These four properties ensure the **sustainability** of open source software. Sustainability implies lower costs over the longer term but, more importantly, reduces the users' reliance on the original vendors of the software. This means that selection criteria that have traditionally been used to ensure the sustainability of software by ensuring the sustainability of the original vendors (e.g. capital, turnover or size requirements) may not be as important and can be reduced for the procurement of open source software. If, for instance, the original vendor goes bankrupt, users can lose all their investments in that vendor's proprietary software. However, if the software is open source, the user can find another vendor to support the software with no legal or technical obstacles.

2.4 "Off-the-shelf" or custom software?

In the public sector, a lot of software is custom-built, or developed in-house. This is partly due to the fairly specific application areas typical to the public sector – for instance, police records management is not a domain with a large private-sector market. According to another EC study[92], about 10% of national public authorities in the EU had or were in a position to release software they owned (custom-built or developed in-house) as open source.

Since such software is generally controlled by the public sector organisation using it, the issues related to open source and open standards are easier to address.

For off-the-shelf software, the *vendor,* not the user, controls the software. Thus, proper procurement procedures are particularly important in the case of off-the-shelf software, in order to help the procuring public agency exercise its control and choice.

92 European Commission DG Information Society and Media, 2008, *Study on the effect on the development of the information society of European public bodies making their own software available as open source.* Available online at http://www.publicsectoross.info

2.5 "Level playing field" or open software?

Current common public procurement practices for software are frequently biased in favour of specific proprietary software vendors. European procurement law may allow for such bias under specific, exceptionally justified situations. However, in practice, neither is this bias exceptional, nor is justification commonly provided.

The scope of this briefing paper is the procurement of open source software. However, many of the principles and methods described in this briefing paper for the procurement of software – whether based on open source or open standards – can be used simply to ensure that procurement of software takes place in a fair environment.

As will be explained in the respective sections below, *any* software procurement should be based on a clear definition of IT architecture, unbiased definition of requirements in *functional* rather than vendor- or brand-based terms and a complete rather than narrow and short-term estimation of costs and benefits. In addition, the methods described for procuring software based on open standards may well be relevant for software in general, justified by cost concerns.

Software acquired after such a process and with such justification may well be proprietary, and will in that case have been properly acquired. Software acquired *without* such a process may well have been acquired improperly and in a biased fashion; if it provides explicit preference for particular vendors, without justification of exceptional circumstances, such acquisition may even be in violation of public procurement regulations.

3 DETERMINING ACQUISITION NEEDS

Public procurement is based on determining needs, identifying the IT architecture in which these must be implemented, translating these into requirements and evaluating available options through the procurement process.

Interestingly, the acquisition of open source software does not necessarily require the use of the public procurement process (i.e. tenders), as purchases of proprietary software and services do. This special case is also discussed below.

3.1 Defining IT architecture

Public sector organisations have architectures that may differ in some respects from private organisations due to differences in their essential goals or principles. Saving costs is a principle that may be common to public and private organisations. Public organisations may differ in that they are obliged to save costs over the very long term - as they are using taxpayer funds and do not need to respond to short term business cycles. However, public organisations often have constraints in the form of budgets that are set for relatively short terms, and therefore still need to balance the short-term and long-term cost savings.

Similarly, transparency is a particularly important principle for public organisations.

The IT architecture needs of public sector organisations are strongly linked to interoperability and open standards. As noted in the European Commission White Paper on ICT Standardisation,[93] "[p]ublic authorities need to be able to define their ICT strategies and architectures, including interoperability between organisations, and will procure ICT systems/services and products or components thereof, that meet their requirements."

Public authorities do not function in a vacuum, and have particularly strong requirements for the interchange of data: between different departments, different organisations, different levels of government – and stakeholders such as businesses and citizens. Public organisations also have obligations towards building sustainable and transparent systems.

93 European Commission, 2009. "White Paper on Modernising ICT Standardisation in the EU - The Way Forward". COM(2009) 324

Interoperability arrangements, when translated into requirements for an IT architecture, justify technical specifications based on open standards. The need to exchange certain data without barriers between citizens and the government, for example, is formalized into a requirement for transparency of those data. This requires transparency in the IT architecture for processes and systems concerned with those data. From this architectural need follows, in terms of technical specifications, the justification for open interoperability standards.

3.2 Determining requirements

Best practice IT procurement is based on defining clear requirements and finding the best match to them. While procurement processes such as calls for tenders, in practice, often ignore this principle to simply specify particular products or even vendors, this is not good practice and may violate procurement regulations. It also makes it more difficult to demonstrate the rationale behind any subsequent acquisition choices.

Requirements can come in a number of forms, which are briefly described below. These are not in any way official categories but are only listed for illustrative purposes.

3.2.1 Functional

Functional requirements describe the purpose for which the IT solution is needed, and the functionality which it is expected to provide. A clear description of functional requirements is essential in order to ensure that procurement follows the principles of transparency and independence, and is pro-competitive and cost effective in the long term. An example of functional requirements would be a detailed description of the functionality that a system for, e.g., maintaining property records is expected to have. Functional requirements should not be defined in IT terms alone, but take in to account the needs to be addressed in the problem domain.

3.2.2 Technical

Technical requirements may also be important, if there are specific constraints or needs regarding the IT architecture and technologies with which the solution must fit. It should be noted that compatibility with previously purchased IT solutions may seem like a very valid technical requirement, but can also be a way of perpetuating the consequences of previous purchasing decisions, perpetuating vendor lock-in and preventing an unbiased procurement based on real organisational needs. Requirements for compatibility with open standards and no proprietary elements, i.e. full compatibility across multiple vendors and producers, increase the freedom of future procurement choices. When compatibility with a previously purchased system requires compatibility with proprietary technologies, it can work against the notion of interoperability across vendors and producers. Such interoperability is essential for the sustainability and long-term cost-effectiveness of software, as already explained above.

In essence, compatibility criteria, when tied to previously purchased proprietary solutions, lock the buyer into that solution indefinitely, making its vendor's one-time win in a single contract a win for a much longer period of future procurements. Since a key principle of public procurement is that a purchase should not have consequences or limit the choice of the buyer after the originally planned lifetime for that purchase, perpetuating such lock-in is a poor procurement practice.

In particular, the effect of previous procurement restricting the choice in future procurement should never last beyond the period foreseen in the original procurement – if a tender was initially for 3 years, the same tender should not result in limiting the choices of a future tender. Such long-term lock-in considerations are often not made in the procurement process, resulting in many tenders calling for branded software from named vendors.

The European Commission itself has reiterated that "[under] the EU public procurement rules, contracting authorities may refer to a brand name to describe a product only when there are no other possible descriptions that are both sufficiently precise and intelligible to potential tenderers"[94]. As a result of this, it is not possible to refer, for instance, to "Intel or equivalent" microprocessors in public tenders.

3.2.3 Business / service model

The needs of the IT architecture and the organisation determine the best form in which an IT solution should be structured, and this includes how it should be paid and accounted for. As a result, certain business models and service models are naturally better fit for a given set of requirements that are determined and defined by a public agency prior to procurement.

This is not, in fact, drastically different from other areas of procurement. A public authority may decide that it wishes to buy a car, or lease it; to commission the construction of a bridge for a fixed fee, or on a build-operate-transfer model.

All these choices involve discrimination between business models, and a preference for some business models over others - simply because a defined set of requirements is better (or only) met by businesses adopting one business model rather than another. Businesses that use a business model that cannot meet the needs of the public agency will naturally lose out. Leasing companies will lose out if an agency's needs are best met by buying rather than leasing cars. This is not against the principles of equal treatment and non-discrimination. However, favouring a particular *business* (a vendor), instead of a *business model*, goes against the principles of equal treatment and non-discrimination. Defining procurement requirements based on particular needs is, however, fully in line with the principles of equal treatment and non-discrimination - even if those needs can only be met by certain business models.

Similarly, when it comes to IT, public authorities are free to choose solutions – and business models - that suit their needs, as long as their needs are clearly justified. Often, such choice - and discrimination - is made by default. For instance, a call for tenders for the purchase of software licences "discriminates" against businesses that do not offer software in the form of a product paid for at the time of purchase through licensing. A call for tenders for software that can be modified, adapted and redistributed by the procuring agency (such software may well meet the open source definition) "discriminates" against businesses who only work on a model based on proprietary control and licensing software for a specified number of users or computers. Of course, businesses may use many different models and are free to adapt their business models to better meet customers' needs. There is no obligation on the part of a public body to adapt its requirements to the preferred business models of particular firms.

3.2.4 Open standards

Open standards in the acquisition of IT may be preferred, or required by policy. With or without an explicit policy at European or national level, open standards may also be preferred or required by policies specific to regions or to particular categories of procurement.

Open standards may take the form of a functional requirement: e.g. it may be an essential function of a new web-based eGovernment service that all citizens have the ability to fully interact with it, without preference to customers of specific software or hardware vendors. Open standards may take the form of technical requirements, for instance when specific open standards are already in use and new acquisitions must work with them.

Open standards may also take the form of requirements that affect business models: e.g., a public authority wishes to have the full freedom to use in perpetuity the data files it creates with new software applications, without being tied to the vendor of that software. It is worth reiterating here the distinction between open standards and open source software. Open standards can be implemented equally well by open source software and proprietary software – and many proprietary software applications implement many open standards.

94 European Commission release reference IP/06/443 dated 4 April 2006; this is also a reference to Directive 2004/18/EC, Article 23.

For the purpose of this briefing paper, the main property of open standards is that the associated intellectual property rights (patents, copyright) are licensed in such a way as to make open source implementations possible.

3.2.5 Open source

As stated previously, there is no EU-wide policy on the procurement of open source software. There are several principles of the functioning of public authorities which may justify the requirement of open source software. The acquisition of open source software can be made on the basis of such justification; a general requirement in a call for tenders for software solutions to be "open source" is not advisable (just as a requirement for “proprietary software” or a brand name would not be advisable) without further details or justification.

As with open standards, open source software can be justified in terms of functional, technical and business model requirements, such as the need to avoid dependence on a single vendor, a need to pay for services rather than license fees, etc. The examples provided above for open standards can to some extent apply as well to open source. Further justifications specific to open source are described below.

As a functional requirement, a public authority may wish to ensure the transparency of government processes. Many of these processes - e.g. for voting systems - are implemented in software, and the only way to ensure its transparency may be to require that the source code be visible for public inspection.

As a technical requirement, a public authority may wish to be able to modify the software (or have any third party of its choice modify it) in the future, in order to work with other software, or have the software adapted to future needs.

As a business requirement, a public authority may wish to be able to distribute the software internally or to other businesses, individuals or agencies with which it interacts, with no additional cost based on the number of users. A public authority may even wish to be able to make adaptations to the software before doing so (or have any third party of its choice make such adaptations). Such requirements, if justified, are perfectly legitimate, and may be effectively requirements for open source software.

In case software redistribution is a requirement, the public authority should determine whether or not it wishes to allow a third party to appropriate the software, i.e. to modify and redistribute it, as if it was proprietary. This will determine the type of licence that should be used in case of redistribution: permissive (in case it is determined that modifications of the software may be made proprietary) or reciprocal (in case it is determined that any redistribution of the software by third parties must remain modifiable and redistributable: this is typically called a “copyleft” license and examples include the GPL and the European Commission’s own EUPL).

Finally, especially in case the public authority wishes to distribute the software to other authorities, business or citizens, a legitimate requirement is to protect the administration from liability, support and warranty obligations relating to redistributed versions of the software.

The above requirements related to redistribution help determine the licence that will be used for this redistribution. As discussed in the later section on ”Defining requirements”, an early determination of this licence (or of a range of authorised licences) is important.

3.3 Examining costs and benefits

Public sector organisations need to keep the public interest in mind, and for procurement this means that public funds should be spent in as cost-effective a way as possible. Freed from the obligation of the short term financial cycles of the private sector, public organisations are also obliged to maximise cost-effectiveness over the very long term. However, with limited, short-term budget cycles, they need to find a good balance between limiting the initial investments and limiting the overall, long term cost.

Although this may be difficult, it is possible to evaluate spending over a long-term horizon to make sure that taxpayers get the best value for their money. It is important to ensure that decisions that look good for the short term do not result in higher expenses and reduced choices over the long term.

3.3.1 Long-term costs

Open source software licences may be available free of charge. This does not mean that the use of open source software is free, of course. Several costs may be involved in the operation of software, including associated hardware, support and maintenance, training and other services. The *exit cost* is also an important consideration, namely the cost incurred in migrating to another IT system. This should properly be accounted for as a cost not of the new system being migrated to, but of the old system being migrated *from*. After all, if the old system were based on open standards, migration would not be as expensive, thus the cost of migration is imposed by the current, old system.

Even if open source software licences are in fact free of charge (and therefore do not even need a call for tenders in order to be acquired, as they can simply be downloaded by a public sector organisation: see the next section), these other costs need to be estimated over the long term. A decision on the software system to be used needs to be made after evaluating all the long term costs associated with the use of that software system.

Similar considerations could be taken into account for the evaluation of proprietary software, which also has requirements for hardware, support, customisation, training and other services. Furthermore, with proprietary software a long term evaluation of costs should include the frequency and necessity of purchasing upgrades.

In a normal procurement process, a pre-defined period is announced at the beginning of the procurement procedure. It is assumed that all costs related to the procured software that will be incurred during that period, such as upgrades, will be taken into account in the evaluation of the bids. A basic assumption of normal public procurement is that at the end of the pre-defined period, the procuring public agency has no contractual obligations towards the original vendor.

However, when software based on proprietary standards and proprietary interfaces is procured, these assumptions of normal public procurement break down. No contractual obligations may exist towards the original vendor beyond the pre-defined lifetime of the original procurement. However, there may be a very high technical and financial cost of moving to a system from another vendor or producer – for example, converting previously created data stored in a proprietary format of the original vendor into another format compatible with other systems. Even acquiring support from another independent vendor, without the support of the original vendor, may have very high costs.

Software is used to create documents, data and customised applications that, in the public sector, have a life-time that may be well beyond the originally announced life-time of the procurement procedure for the software. If the software originally purchased makes it difficult to use the documents, data and customised applications with similar software from other producers, then there is a high cost in terms of changing from the original software to another software - the *exit cost*. With proprietary software this also means there is a high cost in terms of changing from the original vendor to another vendor.

Thus, the assumption of normal procurement procedures, that all costs and obligations relating to procurement are completed after the pre-defined period for which the procurement takes place, appears to fail when applied to software. Contractual obligations do not extend beyond the original procurement period for the software; but the need of the public agency to be able to continue to use its own data and applications means that *technical* obligations come into play, as well. Proprietary standards provide technical obligations that result, in effect, in contractual obligations. This explains why so many public agencies publish tenders for software referring to proprietary software by brand name. They do this because they find the *exit cost* too high, and may simply not quantify it.

Since an essential principle of public sector IT systems is sustainability and independence, the ability to change vendors and systems in the future is essential, and the cost of doing so should be included in the evaluation of the cost of the original software purchase. Hence the term *exit cost*, as these costs are essentially a result of the technical and business model choices of the original software vendor.

The initial selection of proprietary software, if it uses proprietary standards or implements standards in a way that is not exactly the same as software from other producers, can limit future software choices.

As an example, a one-time, presumably competitive acquisition of a proprietary system for web server administration can result in a requirement that all future additions to the web site must be made with the same proprietary system. This not only limits the future choice of the public agency that acquired the software in the first place; it may force citizens who wish to access the public website, as well as businesses and other future contractors developing additions to the public website to become customers of the vendor of the original software acquired by the public agency. In some cases, citizens may not even be able to access such a website without installing a browser from a particular vendor. Such long-term costs of proprietary software are frequently not included in the evaluation process, but are essential for a sustainable, efficient use of public funds.

In brief, long term dependencies on a particular vendor - extending past the boundaries of individual procurement actions - are not good procurement practice and may even be against regulations. Any decision, such as a further procurement action, that reinforces this dependency on a particular vendor, should be avoided, as it will only increase the exit costs.

Note that the argument for the inclusion of exit costs in evaluation is essentially one for open standards, not necessarily open source software[95]. Since exit costs may be hard to quantify at the time of initial procurement, choosing software that works fully with open standards may be a way of avoiding the lock-in effect discussed above.

3.3.2 Long-term benefits (sustainability), additional services

Like costs, benefits should also be evaluated in the long term. Buying new software because it is compatible with previously purchased software may seem to save on migration and training costs. But when this software is proprietary, and is not fully based on protocols and standards that are fully and freely supported by other independent vendors, exit costs and associated costs may greatly increase over the long term, since the agency's dependence on the proprietary vendor is increased as the agency creates more and more systems and data relying on the proprietary software over time. Thus the apparent short term benefit of compatibility is much reduced when considered over the longer term.

Acquiring software that is fully open and sustainable by multiple independent vendors may seem to have less benefits initially, especially if such procurement requires a more detailed study of the market (e.g. for the acquisition of open source software by downloading, or for the identification of appropriate open standards in case of procurement of software that may be proprietary). It may require more detailed procurement specifications, such as functional requirements. And the benefits of having independence and sustainability are not always apparent in the short term. In the long term, however, the ability to change to a new vendor independent of the initial vendor is key to the sustainability and independence of the public agency, and the benefit of such a choice when examined in the long term is thus great.

95 If open source software is being used without open standards, it may implement interfaces and formats that - while not proprietary - are not widely used; it may limit interoperability with other software. However, open source software does not lock the user to the same vendor, and with the source code available, it is possible to have other software adapted to use the protocols implemented, at a cost. Moreover, open source software can often be upgraded at no cost at all - through free downloads - or by any vendors of the procuring agency's choice at a time of the agency's choice.

3.3.3 Total Cost of Ownership (TCO) studies and evaluation

Total Costs of Ownership (TCO) is a term often cited in relation to software purchases. However, there are several different methodologies of computing TCO, and despite the word “total” present in the acronym, few studies include all the long-term costs involved in software purchases, such as the costs of required regular upgrades, or the exit cost of migrating to another software. It is therefore difficult to use TCO studies, or even compare them.

Furthermore, such studies rarely evaluate anything other than quantifiable costs; the benefits of flexibility, independence and transparency, essential to a public organisation, may be qualitative and hard to quantify. Thus, it is advisable to analyse costs and benefits for the needs of the specific public organisation concerned, over the long term, rather than relying on published TCO studies. In particular, if an agency plans to issue a public tender for a solution that will be used for a limited duration of time, it should consider the costs of being able to migrate to a different solution when that duration is complete. Such costs of migration are likely to be higher when the initial solution is proprietary.

3.4 Download or purchase?

Procurement regulations, especially European Directive 2004/18/EC, define when the acquisition of anything, including software, must be put through a public contract, i.e. a formal procurement process such as a call for tenders. As the legal analysis in the Dutch Government's guideline, *The acquisition of (open-source) software,* notes, the acquisition of open source software may not in itself require a call for tenders. This is true especially when this software can be acquired free of charge, i.e. not only free of the licence fee, but also free of any compulsory fees such as for manuals, media or services.

Thus, downloading open source software from Internet repositories free of charge is a means of acquiring software that does not require a public contract. This is true even if the acquiring agency wishes to, in the future, separately acquire paid services or support. For such paid services, of course, a public contract process is required. Regulations *do not* require that the acquisition of software and services be treated as a single acquisition (which would have to be put out to tender), if the software itself can be acquired free of charge, and if this acquisition is *independent of and does not require* those services.

Downloading software free of charge	**Purchasing software**
Large emphasis on market research	Large emphasis on specification
Knowledge to search for the appropriate software to acquire (download) is required by the agency	Bidders provide some of the knowledge, though preparing the tender specifications may also require considerable knowledge
Services must be tendered separately	Software and services can be included in the same tender

4 DOWNLOADING OPEN SOURCE SOFTWARE

When the public agency has decided that open source requirements are particularly important for a specific software acquisition case, the process described in this section can be followed. This process would end in the agency downloading open source software itself, with no fee paid whatsoever. Separately, commercially provided services and support, if required, may be acquired by publishing calls for tender. Note that this process can be abandoned at any point - for instance, if the software cannot be found easily, or evaluated, or once downloaded is found unsuitable for any reason. At that point, the other approach described in the next section can be followed, namely, publishing a call for tender for open source software.

Furthermore, the author recommends the method of downloading open source software *as part of the acquisition process*: downloading software comes *after* all the steps described above, i.e. the determination of requirements, and is simply an alternative to the step of publishing a tender for the supply of software. It is not proposed here as an alternative to the process of managed, well justified and monitored software acquisition.

4.1 Sources of software

Open source software can be redistributed by anyone, so there are naturally many sources for download for most open source applications from the Internet. A number of issues need to be taken into account. Although these are not very different from issues that must be considered while selecting proprietary software, it is worth reiterating them.

4.1.1 Community & language

While selecting proprietary software, it is useful to get to know about the vendor and support network around the software. Although the evaluation of tenders is based on the documents provided with bids, public agencies may already be aware of solutions available on the market thanks to interaction with vendors, reviewing press articles, trade magazines, analysts' reports etc. For open source software, the process of "getting to know" is similar, except that it can be more useful to interact with the community behind a particular open source software application, instead of a particular vendor. As open source software applications can be modified and redistributed, each typically has a *community* behind it, made of different individuals, companies and other institutions - perhaps even public agencies - providing modifications to the software, service and support.

Such a community of users and developers often interacts, and provides some level of mutual support free of charge. Indeed, one of the goals of the EU Open Source Observatory and Repository (OSOR) was to foster such a community for open source software of particular relevance to the European public sector. Similar collaborative platforms for open source software in the European public sector already play this role in countries such as France, Italy, Spain, and Sweden among others.

Open source software is particularly suited to multi-lingual environments, as the freedom to modify and redistribute the software makes it easy for people who speak a particular language to freely add support for it. It is useful to investigate the extent of technical support available for local language versions of the software, as there is often considerable technical support available from user/developer communities.

Finally, there are local support groups for many open source software applications, and it is useful to identify them.

4.1.2 Support & reliability

Open source software, like any software, varies in the level of support available and in the software's reliability. For open source software in particular, communities can provide a fairly high level of support free of charge. This may not be a practical option for any but the smallest public agencies (or, at the other end, larger agencies with significant in-house IT skills). However, this does mean that the software can be downloaded and tested, with the help if required of the supporting communities, before any decision is made on whether or not to deploy it (and perhaps acquire commercial support services).

For many open source software applications, having free support via the community is an order of magnitude quicker and more effective than support by a remote supplier. Also, the community can provide updates to software, making error corrections much quicker than is the case for most proprietary software applications. Indeed, even commercial open source support providers often rely on this free community support, combined with their in-house expertise.

4.1.3 Repositories

Open source software is actually downloaded from repositories of software, or via catalogues. Communities of practice can often be found attached to such repositories.

4.2 Identifying and selecting software

When a number of open source software applications appear to meet an organisation's needs, an evaluation and selection can be performed. This could, first, act as a filter for general reliability and quality as described above, including by taking into consideration issues such as maturity, size of the community, availability of commercial support from various sources, etc. And finally, the selection of the software is based on its matching the previously defined functional requirements.

Functional requirements can be matched to the software documentation - website, software manual, etc. Open source software can simply be downloaded and tested - without deployment, or in pilot deployments - to examine the extent to which it meets functional requirements[96].

Finally, an analysis may be performed of the costs of meeting the functional requirements with the open source software. The solution that is the most cost-effective may be chosen - considering all the various criteria discussed above. If the solutions identified through this process are unsuitable, the procedure of acquisition through downloading can be abandoned, and replaced with a call for tenders for purchasing open source software as described in the next section.

4.3 Tenders for evaluation, support, customisation, services

Downloading software free of charge does not mean there will be no associated costs. While in some cases it may be possible for a public agency to provide all the support for a particular software application in-house, it will often make sense to contract this out. This will naturally require a call for tenders.

To begin with, the process of identification, evaluation and selection of software to download does not have to be performed (entirely) in-house at the public agency. Depending on skills and resources available, it could be useful to have a public contract for some of these tasks. A condition in such calls for tenders may, if justified, exclude the winning bidder from further contracts (such as for services, support) related to the software selected with their assistance, to prevent a conflict of interest and ensure their role as an honest evaluator of open source download choices[97]. Of course, a new tender is not required for every case of software selection. This assistance for evaluation and selection could also be performed by a firm with a pre-existing contract for such on-going consultancy services.

When a final selection has been made for the choice of software to be downloaded, with or without the assistance of a contractor, the software has to be installed, maintained, and supported. Note that downloading software with no contractual arrangements is free of charge, but also means that the software usually comes with minimal warranties. In fact, this is true also for much proprietary software, especially "off-the-shelf" software, where, although many users assume large vendors bear some responsibility for their software, in fact, software licences typically disclaim warranties. As with proprietary software, entering into a service or quality assurance contract of some sort is the main method for a public agency to share some of the responsibility for its use of open source software.

The software may be customised - the ability to be customised extensively is a key advantage of open source software, and customisation may be one reason why open source

96 Of course, proprietary software can also be included in pilot deployments, although if this involves expenditure it may require a formal procurement procedures.
97 An automatic exclusion, according to the case law of the European Court of Justice C-21/03 Fabricom, contravenes the EC Public Procurement directive and such exclusion should be operated only on a case-by-case basis, with bidders “given the opportunity to prove that, in the circumstances of the case, the experience which he has acquired [through the execution of a previous contract such as the selection of technologies or software applications] was not capable of distorting competition”

was selected by the public agency in the first place. For customising public sector software, a paid contractor will usually have to be selected.

For all such additional services, open, competitive calls for tenders should be used to select suppliers. In order to foster the developer community around the selected software, it may be useful to introduce as a criterion in such tenders the level of interaction and contribution the tenderer has within the appropriate community. This can be evaluated through objective metrics such as: the number of posts the vendor (or employees) makes to community online forums; the number of software bug reports filed, bug fixes submitted, or code contributions; the sponsorship and participation of events related to the specific software community such as conferences, workshops, hackathons, etc.

A key property of open source software is that anyone can provide support or other services, depending on their skills. The market is thus fully competitive. No proprietary control or advantage rests with an "owner" or "sponsor", or their dealers and agents. In a call for tender placed for the purchase of specific proprietary software or related services - which works against a competitive market and may violate procurement regulations - only the proprietor itself, or dealers who are necessarily dependent on the proprietor, can bid. In a call for tenders placed for the purchase of services related to a specific open source software system, any independent supplier can bid. In fact, this distinction is similar to the difference that one can find between a tender for the supply of Peugeot cars or services for Peugeot cars (for which only firms dependent on Peugeot can bid), and a tender for fuel and tyre service for a car (for which anyone with no ties to a particular car company can bid).

5 PURCHASING OPEN SOURCE SOFTWARE

Recommended best practice procurement is based on the definition of *functional requirements* - which may include properties that are equivalent to the characteristics of open source software, or the characteristics of open standards.

5.1 Defining requirements

Calls for tenders for open source software - like all calls for tenders - should be based on functional requirements, not on specific products or vendors. Properties of open standards or open source software may be part of these requirements - either as minimum requirements, or as properties that will be favoured.

5.1.1 Functional requirements

The author recommends that the tender specify the function of the software in detail, to ensure transparency and objectivity in the procurement process. The purpose of the software to be acquired and its essential attributes should be described in a vendor-neutral manner. This is a general principle of public procurement; the author focuses here on the *additional* functional requirements relating to the open source nature of the software, and open standards.

5.1.2 Open standards

Provided that this is justifiable due to the interoperability needs of the procuring agency, open standards can be required just as any standards. Open standards can be required either by referring to the open standards by name, or by referring to an official list of open standards. However, if no definition of open standards has been adopted or is applicable to the procuring public agency, nor any officially approved list of open standards can be cited, the standard may have to be defined in terms of functional specifications. The functional properties of “openness”, as described below, could be included among the tender specifications. This way, the openness of standards can be specified as a preference (through the weight given to it in the award criteria), or a requirement (by making it a mandatory in the specifications).

Including open standards requirements or preferences in tender requirements is straightforward: the properties of open standards could be described, together with a justification, if required. Since the justification is part of the essential needs as determined by the public agency, a specific definition of the term 'open standards' is, while useful, less important. For software applications, the needs of a public agency may typically require that:

- *the standard is implementable by all potential providers of equivalent technologies*, ensuring sustainability and full competition with no advantages for some players based on patent or copyright royalties or restricted availability of the technical specifications; in addition, the standard should not discriminate against open source software solutions[98].
- *the development of the standard is open and transparent*, so that the public agency is not dependent on one party for the future of the standard, and may even influence its further development
- *no restriction on re-use*, so that the public agency can be certain that other public or private organisations can use the standard, and so that the use of the standard in open source solutions - which are often not compatible with re-use restrictions - is possible.

Note that these typical public agency needs can be met by standards that fulfil several published open standards definitions.

5.1.3 Open source

As mentioned at the start of this section, it is not good practice to simply state that software should be "open source". Rather, the properties of open source software should be described and justified.

Open source is not part of the technical nature of the software; it applies to the conditions under which the software is provided. Thus, the desired properties of open source could be included in the form of mandatory requirements in the description of the subject matter of the contract, or in the contract documents *(cahier de charges)*. Open source can be included as a preference rather than a mandatory requirement including open source requirements as award criteria.

Including open source requirements is straightforward: the properties of open source could be described, together with a justification, if necessary. The needs of a public agency may typically require that:

- the ownership of the software is transferred to the public agency, with no restrictions on what the agency can do with the software; **OR:**[99]
- *the software may be used for any purpose*, as the public agency does not want to be restricted in how it can use (or allow others to use) the software;
- *the public agency or a third party of its choice may study the source code,* as the public agency wants to be sure of the functioning of the software; alternatively, the public agency may require that any member of the public can study the source code, in order to promote transparency of government processes, or enable other parties to provide support and training associated with the software;
- *the public agency or a third party of its choice may modify the software,* as the public agency does not wish to be dependent on the original vendor for bug-fixes, adaptations and other modifications;

98 The open source non-discrimination requirement is included in the draft version 2.0 of the European Interoperability Framework.

99 Note that some of the requirements below may be met by proprietary software under specific licensing terms, but if all of these requirements are met, the software is by definition open source.

- *the public agency can distribute the software*, with source code and modifications, to anyone of its choice and provide recipients with the same abilities to use, study, modify and redistribute, because the public agency needs to ensure that citizens and firms and other agencies that access its services using the software or variants of the software do not need to become customers of the original vendor in order to do so; for example, a national administration may wish to be able to pass on the software, without extra costs, to other administrations at the local, regional, national or European levels.

When supported by an official policy at European, national or local level, such requirements may not need explicit justification in each tender. Even if there is no official policy, such criteria need only to be *justifiable* - i.e. if questions are raised - rather than justified in each tender. But there is no harm providing explicit justification and references, and it is always a good practice to (briefly) explain why certain criteria are present. For instance, the explanation for the Dutch government's preference for open source software is the "promotion of a level playing field in the software market and promotion of innovation and the economy".

5.1.4 Open source for redistribution

When procuring software for the purpose of potential redistribution it is important to clarify specific redistribution conditions for the component being procured, to ensure that the public authority does not face licensing difficulties combining that component with other, separately acquired software, for further redistribution.

To address such cases, Spain has adopted Royal Decree 4/2010 implementing the National Interoperability Framework planned in the eGovernment Law of 11/2007.[100] According to RD Article 16.1, the licensing conditions of applications owned by Public Administrations that can be made available for other Public Administrations or for the citizens, must:

- allow the free use/reuse of these applications;
- exclude the software appropriation by a third party;
- protect the administration from liability, support and warranty obligations.

Therefore, if the distribution is decided, it must be under open source conditions (combined with strong “copyleft” conditions for avoiding the exclusive appropriation that would happen if the software could become proprietary).

This means that, by default (and by choice, when it is appropriate), the Spanish administration will distribute its software under the terms of the European Union Public Licence (the OSI approved licence which has the same value in 22 European languages)[101].

5.2 Other requirements

In addition to technical, functional requirements and the non-technical properties of open source and open standards, calls for tender typically contain also other criteria for awards and for determining the eligibility of bidders.

One property of open source software that distinguishes it from proprietary software is that small, innovative companies, limited only by their skills and abilities rather than their dependence on the software proprietor, can provide it on an equal basis. However, small companies may have difficulties meeting stringent eligibility criteria with regard to financial sustainability.

100 The Royal Decree (text in Spanish) is published at: http://www.csae.mpr.es/csi/pg5e41_ingles.html. On ePractice see the presentation and comments from Miguel A. Amutio (in English) http://www.epractice.eu/en/cases/eni and comments from P-E. Schmitz on OSOR http://www.osor.eu/communities/eupl/blog/impact-of-the-spanish-royal-decree-4-2010-of-8-january-2010-1.

101 Similar provisions can be found in other recent policies in Member States, such as the 1 June 2010 directive of the Government of Malta enabling, where appropriate, the distribution of its public sector software under the EUPL - http://ictpolicies.gov.mt/docs/GMICT_D_0097_Open_Source_Software_v1.0.pdf

Selection criteria for financial sustainability (minimum turnover, capital) should be in proportion to the scope of the tender[102]. The main justification for financial sustainability criteria for software is to ensure that the supplier will be able to provide support as long as the software is being used.

With open source, the availability of the source code assures interoperability, and there is no dependence on the original supplier. If the original supplier goes out of business, the software can still be maintained by others; if others are not maintaining the software, the public agency can hire a third party maintainer. This increased sustainability of open source is justification for lowering the financial sustainability requirements, or lowering their weight in the selection process for tenders for open source software.

5.2.1 Community interaction and contribution

One of open source software's main strengths is that the development process, at its best, involves a community of several firms, individuals and other contributors. Contribution is not limited to actual writing of lines of code, and extends to, for instance, providing detailed reports of requirements and issues.

Thus, it may be useful to include the level of interaction and contribution the tenderer has within the appropriate community as an award criterion in tenders or as an element demonstrating process quality during the execution of the contract.

5.3 Tender selection

Bids responding to the call for tenders must be evaluated, and the best offer chosen by the lowest price, or the best value for money as determined by the weighted award criteria.

In case of a preferential policy regarding open source - such as with the Dutch government's "preference for open-source software in the case of equal suitability" - if bids have the same price (in case of a lowest price tender) or the same value-for-money, the open source bid is selected. Note that any such preferences must be justified in terms of the functional, technical specifications and must not create obstacles to the "*opening up of public procurement to competition*"[103]. As open source requirements are put in place in order to meet the needs of the procurement body and do not favour specific vendors, in contrast to procurement procedures requiring specific named brands of software, in general they add to the opening up of procurement to competition.

Any such preferential policy related to open source does not really affect the tender process described in this section, as the likelihood of exactly the same evaluation of two bids is probably not high. Moreover, the inclusion of open source requirements as part of the tender requirements is independent of any policy regarding open source in procurement; *it requires no preferential policies and works within any procurement procedures.*

102 Directive 2004/18/EC, Article 44(2).
103 Directive 2004/18/EC Article 23(8)

Legal aspects of free and open source software in procurement: national case studies

Philippe Laurent, University of Namur

ABSTRACT

Member States' public authorities are increasingly interested in the advantages of procuring free and open source software. Some of them have already adopted different strategies to raise awareness, to level the playing field or even to establish positive discrimination for such permissively licensed software. This briefing paper aims at illustrating the current political and legislative trends by observing cases from the Netherlands, Italy, Spain, the United Kingdom, Belgium and France

CONTENT

EXECUTIVE SUMMARY

Background

Free and open source software (hereinafter referred to as "FOSS") is software licensed under permissive terms, which enable the licensee to use, reproduce, modify and re-distribute the software and its modifications at will.

This peculiar licensing scheme harmoniously fits the general public procurement principles of transparency, flexibility, independence, interoperability, sustainability and cost-effectiveness. Nonetheless, it has been observed that public procurement practices often tend to disadvantage the adoption of FOSS. Some policy makers have therefore elaborated diverse strategies to fix the problem, such as:

- the Dutch government with its NOIV action plan,
- the Piedmont Regional Council with its Act on software pluralism in the administration,
- the Spanish lawmaker with its National Interoperability Framework,
- the UK government with its ICT Government Strategy,
- Walloon municipalities with the creation of an IT public company called IMIO, and
- the French Prime Minister with his Circular on the use of FOSS in administrations.

Aims

This briefing paper aims at illustrating how Member States' public administrations (hereinafter 'PAs') at different administrative levels have implemented government strategies and legislative texts to raise awareness, to level the playing field or even to establish positive discrimination for such permissively licensed software in procurement contexts.

It also aims at comparing these initiatives so as to identify some lessons that can be drawn from the different experiences.

KEY FINDINGS

- The different initiatives analysed are not at the same stage of development and are very diverse in terms of scope, scale, means and ambitions, which renders precise comparison difficult.
- All the policy makers behind the analysed strategies were aware of the potential and advantages of FOSS. Software reuse and costs reduction seem however to be the two main incentives that generally triggered the initiatives.
- The degree of success of the different initiatives is very variable.
- The current public procurement regulatory framework as such does not seem to constitute a hindrance to the adoption of FOSS by administrations. It provides ways to develop practices aimed at levelling the playing field or granting preference to the procurement of FOSS.
- Contracting authorities seem however to show different degrees of resistance, which is motivated by multiple factors that tend to be overlooked.

1. BACKGROUND

Public services have become increasingly and irreversibly dependent upon information and communication technologies. Many if not all administrations, at any level, have more or less incorporated ICT into their operations. Whereas some of them mainly use simple mainstream systems such as word processors, spreadsheets applications, emails, Internet browsers, etc., other public services use complex – and usually highly if not totally customised – database systems and information systems. Software represents a qualitatively and quantitatively essential part of such systems. Accordingly, software procurement has become a key element in the general administration governance, which has a direct influence on the quality and effectiveness of its services.

The law regulates public tendering in order to ensure that economic operators are equally treated and in order to safeguard the financial, economical and operational interests of the contracting authority, which can be associated with the public interest[104]. Directive 2004/18/EC on the coordination of procedures for the award of public works contracts, public supply contracts and public service contracts[105] provides only for a general legal framework establishing global principles such as transparency and non-discrimination. National and local lawmakers and public administrations therefore benefit from a significant leeway and may take important decisions as regards public ICT and software procurement policies.

[104] D. De Roy, "L'irruption du logiciel libre dans le secteur public. A la découverte d'une actualité fort ancienne", in *Les logiciels libres face au droit*, Bruxelles, Bruylant, 2005, p.200.

[105] Directive 2004/18/EC of the European Parliament and of the Council of 31 March 2004 on the coordination of procedures for the award of public works contracts, public supply contracts and public service contracts, *OJ* L 134, 30 April 2004, pp. 114–240.

Procurement practices have often been criticised for discriminating against FOSS or excluding it from competition. Such exclusion does not always happen voluntarily, but often results from misunderstanding or ignorance of the FOSS licences mechanisms and the associated business models. For instance, besides the all too common requirements of, or references to, proprietary trademarks or technologies, award criteria requiring the bidder to be the owner of the copyrights or referring to the "purchase" of software licences are equally detrimental to FOSS-based bids[106].

Understanding FOSS and the business models which have been developed around it is the first prerequisite to improve procurement practices. Traditional "proprietary software"[107] business models are usually based on the exclusive exploitation of intellectual property and the "sale" of licences limiting the scale and extent of software usage. FOSS is, on the contrary, based upon a permissive licensing system coupled with an unrestricted access to the source code, which enables the licensee to use, reproduce, modify and re-distribute the software (and its modifications) at will[108]. In addition to being very permissive, FOSS licences are royalty-free: licensors are not remunerated in exchange of the given authorisation.

FOSS licensing uses intellectual property (normally copyright) in a versatile way, not to monopolise technology and reap royalties, but to foster creation on an open and collaborative basis. This very peculiar licensing scheme is sometimes described as a way to reconstruct virtual commons[109], namely open to all and non-exclusive resources. Such peculiar licensing scheme has challenged the traditional business logic and has given birth to alternative models, which are generally more focussed on the provision of services (around the shared resources) than the selling of products (created on the basis of monopolistic rights on the resource).

Like the European Union[110], many Member States and administrations at national, regional or local level are receptive towards the potential and advantages of FOSS, which fit the general public procurement principles of transparency, flexibility, independence, interoperability, sustainability and cost-effectiveness[111]. Accordingly, some national and local governments have taken very diverse measures in order to promote the use of FOSS in the administrations and to better adapt their procurement policies so as to take into account FOSS specificities and to open the competition to FOSS-oriented bids.

The Legal Affairs Committee of the European Parliament has decided to hold a workshop on the legal aspects of the use of FOSS, in which the legal aspects of procurement will also be outlined. In this context, the Committee requested an *ad hoc* briefing paper identifying and summarising relevant national experiences at different levels (national, regional or local) to illustrate the current trends regarding the procurement and use of FOSS by public administrations within the EU.

---

[106] "*For instance, a call for tenders for the purchase of software licences "discriminates" against businesses that do not offer software as a product paid for at the time of purchase through licensing*". IDABC, Guideline on public procurement of Open Source Software, available at https://joinup.ec.europa.eu/sites/default/files/studies/OSS-procurement-guideline-public-final-June2010-EUPL-FINAL.pdf.

[107] It is common to use the term "proprietary" software to refer to software that is not licensed under a FOSS licence but governed by restrictive terms, and the use of which requires the payment of royalties.

[108] See "The Free Software Definition", available on the FSF official website, http://www.gnu.org/philosophy/free-sw.html and "The Open Source Definition", available on the OSI official website, http://opensource.org/docs/osd.

[109] PH. LAURENT, "Free and Open Source Software Licensing: A reference for the reconstruction of "virtual commons?" to be published in the proceedings of the Conference for the 30th Anniversary of the CRID, which took place in Namur from the 20th to the 22th of January 2010, available at http://www.crid.be/pdf/public/7133.pdf.

[110] In recent years, the European Union has paid increasing attention to the potential of free and open source software. Already in its 2002 Communication "eEurope 2005: An information society for all," [COM(2002) 263 final, 28.5.2002], the European Commission stated that it intended to promote the use of open standards and of open source software. As from 2006, the IDABC and ISA programmes of the European Commission are actively promoting the use of FOSS. The European Commission even created and stewards the OSI certified European Union Public Licence (EUPL). Interest in free and open source software has again increased after the European Commission published, in September 2012, the Communication "Unleashing the potential of cloud computing in Europe"[COM(2012) 529 final, 27.9.2012].

[111] IDABC, "Guideline on public procurement of Open Source Software", March 2010 (revised June 2010), available at https://joinup.ec.europa.eu/sites/default/files/studies/OSS-procurement-guideline-public-final-June2010-EUPL-FINAL.pdf.

2. METHODOLOGY

This paper reports on a selection, analysis and comparison of different national and local initiatives that have been implemented in order to improve procurement practices and to invite administrations to better consider FOSS in software procurements.

This paper is far from being exhaustive and aims exclusively at illustrating some of the many approaches adopted at different administrative levels in order to give a first insight into the problems confronted and/or the results achieved. The selection has also been made considering the direct accessibility of information and the purposes and limits of this briefing paper.

In order to facilitate the observation of the different initiatives, some comparison points have been identified.

The administrative level (national, regional or local) where the decision has been taken and implemented is the first element of comparison.

The initiatives are classified into two categories, depending on their nature: legislation (law or decree) or policy (programme, action or any other initiative from an executive body).

For each case, the relevant legal background has been globally analysed. The paper describes the fixed objectives and how FOSS is being dealt with in such a legal framework to reach these objectives.

Three types (or levels) of objectives are identified and serve also as a general comparison point: raising awareness on FOSS, ensuring non-discriminatory treatment, and actively encouraging or preferring FOSS procurement. If this third objective is upheld, and where possible, the question whether or not the initiative addresses the issue of the selection and assessment of awarding criteria is briefly tackled.

Where possible, information on the reception and effectiveness of the analysed solutions has also been gathered and assessed with regard to the following questions, where relevant: how effective different solutions have proven to be in practice in enabling FOSS procurement, how they have been applied by administrations and/or the courts (some of these initiatives have been challenged before court), and what types of licences are involved.

One must finally note that, although open standards and FOSS are close concepts that are usually addressed jointly to elaborate effective procurement strategies, this briefing paper only focuses on the procurement of FOSS.

3. EXPERIENCES

3.1 Netherlands: NOIV action plan (2007-2011)

3.1.1 General presentation

The NOIV programme was an action plan[112] that aimed at accelerating the use of open standards and open source software[113] within the national government, subsidiary government bodies and the public and semi-public sector.

It had been adopted by the Dutch Government and had been implemented during the governmental session (from 2007 to 2011) by a subdivision of ICTU (ICT-Uitvoeringsorganisatie), an organisation established by the Ministry of the Interior and Kingdom Relations, and the Association of Municipalities.

The main objectives of this action plan were:

[112] Available in English at https://www.ictu.nl/archief/noiv.nl/files/2009/12/Action_plan_english.pdf.
[113] "Open source software" is defined by reference to the OSI definition. See "De stand van zaken van het open source software beleid van de Rijksoverheid", available at http://www.ictu.nl/archief/noiv.nl/documenten-en-publicaties/doc/het-open-source-beleid-van-de-rijksoverheid/index.html#more-7119.

- to increase interoperability by accelerating the use of open standards,
- to reduce supplier-dependence through a faster introduction of open source software and open standards, and
- to promote a level playing field in the software market, by promoting innovation and the economy by forceful stimulation of the use of open source software, and by giving preference to open source software during the process of IT acquisition.

Accordingly, three procurement principles were upheld by NOIV:

- Open source is not mandatory, but its use should be strongly encouraged,
- Open source software should be preferred if it is equally suitable, and
- Providers of open source software should have the same opportunities as providers of closed source software.

In order to foster the use of FOSS by administrations at any level, some action lines had to be followed. Some of them can be summarised as follows:

- A programme office within ICTU had been set up to support actively the action lines. The office provided guidance, result-oriented advices and customised practical support to the administrations. It also conducted measurements to monitor the progress of the actions. A ranking has been maintained and prizes were offered annually for the Most Open Public Organisation.
- At national level, meetings with businesses, suppliers and various government target groups were organised to explain the plans and to reach practical agreements for their implementation.
- Any administration had to develop an implementation strategy for tendering, purchasing and using open source software. A fixed deadline (January 2009) was adopted for the ministries.
- The Government was also to encourage the use of open source software in a European context.
- The Government also intended to investigate to what extent all software under its control or developed on its order could in principle be released under an open source software licence. To that end, it showed specific marks of interest towards the European Union Public Licence (EUPL). The Government realised that such objective could mean that it would have to make tenders for development of software conditional on its obtaining of the intellectual property for the software developed.

NOIV was therefore a general framework set up at national level to foster the development and adoption of pro-FOSS procurement strategies in any Dutch administration.

NOIV clearly stated that the procurement rules are not applicable when freely downloading FOSS. It noted, however, that administrations should select downloadable FOSS with care and according to strict procedures. Notwithstanding this, the procurement of services around such selected software (such as deployment, maintenance, customisation or support services) must be done conforming to the classical rules, bearing in mind that "open source software is provider-independent"[114].

Amongst the documents issued by the NOIV, a guideline has been published that provides examples of award criteria to be added in the calls for tender[115] and which can be summarised as follows:

[114] « Download open source software », available at https://www.ictu.nl/archief/wiki.noiv.nl/xwiki/bin/view/NOiV/Downloaden%2Bvan%2BOpen%2BSource%2Bsoftware.html.

[115] "Modelteksten voor open voorkeur in een (Europese) aanbesteding", NOIV – November 2010, available at http://www.ictu.nl/archief/noiv.nl/files/2010/11/NOiV_Modelteksten_voor_open_voorkeur_in_een_aanbesteding.pdf.

- the involvement of the bidder in a FOSS development community (the criterion is the number of developers who are members of the community);
- the participation of the bidder in the development of the provided software (the criterion is the percentage of code that has been sent by the bidder as contribution(s) to the project);
- the adoption by the bidder of a procedure ensuring the origin of the source code that he provides;
- the bidder's experience with W3C[116] web content guidelines;
- the adherence of the bidder to a "open providers manifest" (issued by NOIV);
- the database independence of the software;
- the platform independence of a user interface;
- whether there is a large number of maintenance service providers available;
- the granting of rights (by way of a licence) to modify, to further develop and to redistribute at will the source code of the software;
- the existence of an independent and freely accessible community of developers who are involved in the development of the software (and of future versions thereof);
- whether the applications can be deployed on a diversity of different hardware and operating systems; and
- the priority given to open standards.

As regards custom made code, the guideline explains that, instead of requiring the transfer of IP, the call could provide that the code is delivered under the EUPL or another OSI certified licence.

3.1.2 Results

The NOIV office has yearly monitored the progress in open standards and FOSS adoption, and released interesting and very detailed reports[117]. In general, open standards adoption seems a higher priority than FOSS adoption.

Conforming to what was expected from them in the action lines, all the ministries reported having adopted a procurement strategy. NOIV noticed, however, that the ministries did not seem to discriminate in favour of open source but neutrally aimed at "procuring the best software". It further noticed that awareness seemed to have been raised, but that the procurement practices could nonetheless be improved[118].

Mathieu Paapst (ex-member of the NOIV office) is even more pessimistic about the results of the programme after a survey conducted on 80 Dutch calls for tender published between January and June 2010. To the question whether a policy like the action plan NOIV influences behaviour regarding the practice of public tenders, he answers that *"despite the desired affirmative action for Free/Libre and Open Source Software, in almost half (47.5%) of the tenders there is* [according to the way the terms are drafted] *a preference for closed source vendors or products. Because of this preference vendors of FLOSS products are not given a fair chance to win the bid. There is no level playing field in the software market and government buyers arguably do not act according to the EU treaty principles of equal treatment, non-discrimination and transparency"* [119]. Mathieu Paapst noticed that 22 tenders out of 80 mentioned a preference for FOSS, out of which 15 only provided that such preference would actually result in a reward of extra points under the weightings applied to the award criteria.

[116] The W3C is the main international standards organization for the World Wide Web.
[117] The more recent monitoring report that we found is the "Monitor NOIV 2010" of January/February 2011, which is available at http://www.ictu.nl/archief/noiv.nl/files/2011/06/NOiVmonitor2011.pdf.
[118] "De stand van zaken van het open source software beleid van de Rijksoverheid", *op. cit.*
[119] M. PAAPST, "Affirmative action in procurement for open standards and FLOSS", *IFOSSLR*, 2010, vol.2, issue 2, available at http://www.ifosslr.org/ifosslr/article/view/41/76.

3.1.3 Features

• NATURE:	Policy (official programme of the government)
• DECISION LEVEL:	National (Dutch government)
• ACTION LEVEL:	Any level (central government, provinces, local authorities)
	Any governmental institution (education, healthcare, social security)
• OBJECTIVES:	Raising awareness of FOSS
	Promoting a level playing field
	Giving preference to open source software, if equally suitable
• MEASURES TAKEN:	Promotion of FOSS
	Creation of a support office (which issued many guidelines)
	Guidance and support
	Guidelines on award criteria
• LICENSING:	Procured software should be under an open source licence as defined by the OSI
	The EUPL is considered when an administration contemplates to license its own software
• EFFECTIVENESS:	Awareness has increased.
	As regards the objective to level the playing field, practices do not seem to have been satisfactorily improved.
	Positive discrimination has in general not been adopted.
	A minority of administrations has, however, adopted FOSS oriented awarding criteria.

3.2 Italy: Piedmont Region's Act of 2009 and beyond

3.2.1 General presentation

Italy is an unitary state, organised in such a way that many matters are reserved to the State, but regions can nonetheless adopt specific laws on their own internal functioning. In 2009, the main national law that governed software procurement was neutral as regards the nature of the procured software, FOSS being an option amongst others[120].

As regions have the power to adopt more detailed rules with regard to public procurement, the Piedmont Regional Council passed, on 26 March 2009, an Act establishing "*rules on software pluralism, on the adoption and the diffusion of free software and on the portability of digital documents in the public administration*"[121].

The aim of the region was to give priority to FOSS.

This is clearly reflected in the far reaching provisions of the adopted law, which provides, amongst others, the following:

- The region uses software applications whose source code is available and which it can freely modify to adapt them to its needs.
- Except for software already in use, the region must preferentially procure Free Software and software whose source code is verifiable by end users.
- When procuring software, the region must carry out a technical and economic comparative assessment among the different solutions available on the market. In

[120] C. PIANA, "Italian Constitutional Court gives way to Free-Software friendly laws", *IFOSSLR*, 2010, vol.2, issue 1, available at http://www.ifosslr.org/ifosslr/article/view/38.

[121] Legge regionale n. 9 del 26 marzo 2009, Norme in materia di pluralismo informatico, sull'adozione e la diffusione del software libero e sulla portabilità dei documenti informatici nella pubblica amministrazione, (B.U. 02 Aprile 2009, n. 13), available at http://arianna.consiglioregionale.piemonte.it/ariaint/TESTO%3FLAYOUT=PRESENTAZIONE&TIPODOC=LEGGI&LEGGE=9&LEGGEANNO=2009

doing so, the region takes into account the total cost of ownership of each solution, the exit costs, but also the potential interest that other administrations could see in reusing the software and its interoperability potentials.

- If the region decides to use proprietary software, it must justify the reasons for such a choice.
- The region makes available - as free software - the computer programs that are developed on the basis of its own specifications and that are entirely financed by public funds.

3.2.2 Results

This initiative was acclaimed by FOSS advocates, not however by the national government. Indeed, the Italian government deemed that by adopting such law, the Piedmont region had exceeded its authority. The national government therefore lodged a claim before the Constitutional Court, raising two main grounds for annulment. The Constitutional Court issued a decision on 23 March 2010[122].

The first ground for annulment, based on the fact that copyright law is a matter that is reserved to the central state, was upheld by the court. The corresponding illegal provision was declared illegal.

The second ground for annulment was more specifically aiming at the pro-FOSS provisions of the regional law. The Italian government alleged that any provision favouring FOSS adoption would be in conflict with the national laws on competition, as it would discriminate against the proprietary software industry[123].

This argument was not upheld by the Constitutional Court, which answered the argument as follows:

> "*The choice is not an exclusive one, but just preferential and requires a comparative evaluation, as is confirmed by the reference to the possibility to use proprietary formats […] under the condition that in such case the Region shall provide motives of its choice […].*
>
> *Finally, it must be once more reminded that the concepts of free software and software with inspectable code are not notions concerning a given technology, brand or product, instead they express a legal characteristic. At the end of the day, what discriminates between free and proprietary software is the different legal arrangement of interest (licence) upon which the right of using the program is based; and the choice concerning the adoption of one or the other contractual regime belongs to the will of the user.*
>
> *It follows that the damage to competition feared by the counsel of the State with regard to the law in question, is not envisaged"*[124].

Marco Ciurcina, who was at that time president of the ASSOLI (Associazione per il Software Libero), assessed that the Constitutional Court's decision, which was welcomed by the Association[125], would make it easier for other administrations to adopt similar laws[126].

This premonition proved to be right, as in 2012, the national Digital Administration Code (Codice dell'amministrazione digitale[127]) was modified twice in order to establish a preference for FOSS in the public administrations.

[122] Sentenza 122/2010, *Corte Costituzionale della Repubblica Italiana*, available at http://www.cortecostituzionale.it/actionPronuncia.do.
[123] According to Roberto Di Cosmo, this argument would have been inspired by proprietary software lobby. See R. Di Cosmo, "Constitutional Court in Italy rules out anti-free-software lobbyist arguments...", 30 March 2010, available at http://www.dicosmo.org/MyOpinions/index.php/2010/03/30/100-constitutional-court-in-italy-rules-out-anti-free-software-lobbyist-arguments.
[124] As translated by C. Piana in "Italian Constitutional Court gives way to Free-Software friendly laws", *op. cit*.
[125] "A landmark decision of the Italian Constitutional Court: granting preference to free software is lawful", available at http://www.softwarelibero.it/corte-costituzionale-favorisce-softwarelibero_en.
[126] Joinup News of 30 March 2010, "IT: Constitutional court says administrations can favour open source", available at https://joinup.ec.europa.eu/news/it-constitutional-court-says-administrations-can-favour-open-source.

Article 68, part 1 and 2 of the Code read as follows:

"1) In accordance with the principles of economy and efficiency, return on investment, reuse and technological neutrality, public administrations must procure computer programs or parts thereof as a result of a comparative assessment of technical and economic aspects between the following solutions available on the market:

> *a) develop a solution internally;*
>
> *b) reuse a solution developed internally or by another public administration;*
>
> *c) adopt a free/open source solution;*
>
> *d) use a cloud computing service;*
>
> *e) obtain a proprietary license of use;*
>
> *f) a combination of the above.*

1-bis) For this purpose, before procuring, the public administration (in accordance with the procedures set out in the Legislative Decree 12 April 2006, n. 163) makes a comparative assessment of the available solutions, based on the following criteria:

> *a) total cost of the program or solution (such as acquisition price, implementation, maintenance and support);*
>
> *b) level of use of data formats, open interfaces and open standards which are capable of ensuring the interoperability and technical cooperation between the various information systems within the public administration;*
>
> *c) the supplier's guarantees on security levels, on compliance with the rules on personal data protection, on service levels [,] taking into account the type of software obtained.*

1-ter) In the event that the comparative assessment of technical and economic aspects, in accordance with these criteria of paragraph 1-bis, demonstrates the impossibility to adopt an already available solution, or a free/open source solution, as well as to meet the requirements, the procurement of paid-for proprietary software products is allowed. The assessment referred to in this subparagraph shall be made according to the procedures and the criteria set out by the Agenzia per l'Italia Digitale, which, when requested by interested parties, also expresses opinions about the compliance with them.

2) In the preparation or acquisition of computer programs, public administrations, whenever possible, must adopt solutions which are: modular; based on functional systems disclosed as stated by Article 70; able to ensure the interoperability and technical cooperation; able to allow the representation of data and documents in multiple formats, including at least one open-ended (unless there are justifiable and exceptional needs).

2-bis) The public administrations shall promptly notify the Agenzia per l'Italia digitale the adoption of any computer applications and technological and organizational practices they adopted, providing all relevant information for the full of the solutions and the obtained results, in order to favour the reuse and the wider dissemination of best practices" [128].

Even though the provisions of § 1-bis are not unambiguous and need interpretation, and although the role of the *Agenzia per l'Italia digitale* could have been further detailed, it is clear from § 1-ter that the procurement of FOSS must be preferred to proprietary software. Not only would procuring FOSS comply with the order of priority as established in the law, but it would also allow administrations to reuse and share software amongst them, which seems to be the final goal of the Italian lawmaker.

[127] Available at http://www.digitpa.gov.it/amministrazione-digitale/CAD-testo-vigente.

[128] As translated by S. ALIPRANDI & C. PIANA in "FOSS in the Italian public administration: fundamental law principles", *IFOSSLR*, 2012, vol.5, issue 1, available at http://www.ifosslr.org/ifosslr/article/view/84.

According to Simone Aliprandi and Carlo Piana, "*The decision was made in a dire situation of the national economy and inspired by practical reasons (spending review) rather than idealistic ones. It seems however a new direction that can hardly be changed. Only it can be made less compelling by a slack implementation, if not outright non compliance. Vigilance is therefore required*"[129].

3.2.3 Features of the Piedmont Region's Act

• ACTION:	Legislation
• DECISION LEVEL:	Local (Piedmont region)
• ACTION LEVEL:	Local (Piedmont region)
• OBJECTIVE:	Favouring the procurement, sharing and re-use of FOSS by the administrations
• MEASURES TAKEN:	Adoption of a law establishing procurement rules
• LICENSING:	Not specified (reference to free software)
• EFFECTIVENESS:	The initiative has been successful and survived a challenge before the Constitutional Court. A couple of years after the Constitutional Court's decision, at national level, the Code for the Digital Administration has been modified to favour FOSS and promote sharing and re-use of software. How the amended code will be concretely applied remains uncertain.

3.3. Spain: National Interoperability Framework

3.3.1 General presentation

The Spanish Citizens Electronic Access to Public Services Act (eGov Law 11/2007)[130] grants citizens the right to interact with the public administration by electronic means. The law regulates the basic aspects of IT use, but also the cooperation between administrations and the reuse and transfer of applications amongst them (articles 45 and 46).

In application of article 42 of the Law, the Royal Decree 4/2010[131] implements the Spanish National Interoperability Framework, and contains several provisions (articles 16 and 17) aiming at fostering the use of FOSS in the public sector.

Article 45 of the Law provides that when public administrations are owners of IP rights on their applications, they may[132] make them available to any other public administration without compensation and without the need of an agreement. These applications can be declared "open source" when this allows a better transparency in the functioning of the public administration or when this fosters the incorporation of the citizens in the information society.

Article 16 of the Royal Decree does not oblige public administrations to redistribute their applications to other administrations and citizens, but if they do, they must take into account that the aim is to allow the use and the reuse of the software, as well as the protection against its exclusive appropriation by a third party. The transferor must however protect itself from liability, support and warranty obligations. The provision details the licensing conditions, which must ensure that

- the software can be executed for any purpose,
- the source code is available,

[129] *Idem.*
[130] Ley 11/2007, de 22 de junio, de acceso electrónico de los ciudadanos a los Servicios Públicos, available at https://www.boe.es/buscar/doc.php?id=BOE-A-2007-12352.
[131] Real Decreto 4/2010, de 8 de enero, por el que se regula el Esquema Nacional de Interoperabilidad en el ámbito de la Administración Electrónica, available at http://www.boe.es/buscar/doc.php?id=BOE-A-2010-1331.
[132] The provision reads as follows: "*Las administraciones titulares de los derechos de propiedad intelectual de aplicaciones, desarrolladas por sus servicios o cuyo desarrollo haya sido objeto de contratación,* podrán *ponerlas a disposición de cualquier Administración sin contraprestación y sin necesidad de convenio*".

- the software can be modified or improved, and
- the software can be redistributed to other users with or without changes, on the condition that the derived work keeps these four "guarantees".

The last condition entails the use of a copyleft licence, namely a licence which provides a specific clause that, generally speaking, obliges anyone who redistributes the software, with or without changes, to redistribute it under the same licence[133].

Outstandingly, the last paragraph of article 16 provides that

> *"To this end, the application of the European Union Public Licence will be sought, without prejudice of other licences that can guarantee the same rights stated in the* [previous] *paragraphs* […]"[134].

The EUPL is therefore set by law as the licence "by default"[135] to be required in procurements, and which has been pre-validated by the lawmaker as being compliant with the 4 conditions set forth above. If the public administration wants to use another licence, it has to check whether the contemplated licence meets the same conditions.

Article 46 of the Law provides that public administrations must keep updated registries of applications for free reuse, in cooperation with a Technology Transfer Centre that is set up and run by the General State Administration, and conforming to the principles provided by the National Interoperability Framework. Article 17 of the Royal Decree further provides that public administrations have to take into account the solutions freely reusable available in those registries and which could meet (totally or partially) the requirements of the new systems and services, and consider the improvement and update of the solutions already implemented. In order to optimise the sharing and collaborative process, the ongoing development should be published in the registries at an early stage, without waiting for the code to be finalised.

"Reuse" is therefore the catchword in Spain as regards ICT public procurement, and FOSS seems to be perceived as a key element for achieving this goal. However, the Royal Decree does not establish any preference for the acquisition of software products based on FOSS.

3.3.2 Results

The Technology Transfer Centre has been created[136]. It keeps and maintains the repository of reusable software, which is connected to several forges[137] operated by autonomous communities (Andalusia, Catalonia and Extramadura) and to the European JOINUP platform (operated by the ISA programme)[138].

The Centre is functioning hands in hands with the CENATIC (*Centro Nacional de Referencia de Aplicación de las TIC basadas en fuentes abiertas*), which is a very active centre created by the Spanish Government and which raises awareness on and promotes the usage of FOSS in any sector, with a special focus on, amongst others, the public administrations. CENATIC organises a national observatory of free software[139], which has released the

[133] The GPL is the most famous copyleft licence. There are however many types of "copyleft effects" which cannot be further described in the present briefing paper. See for instance PH. LAURENT, "Free and Open Source Software Licensing: A reference for the reconstruction of "virtual commons?", *op. cit.*

[134] The provision reads as follows: "*Para este fin se procurará la aplicación de la Licencia Pública de la Unión Europea, sin perjuicio de otras licencias que garanticen los mismos derechos expuestos en los apartados 1 y 2*".

[135] See P.E. SCHMITZ, « Impact of the Spanish Royal Decree 4/2010 of 8 January 2010 », available at http://joinup.ec.europa.eu/community/eupl/news/impact-spanish-royal-decree-4/2010-8-january-2010.

[136] Information on the Technology Transfer Center is available at http://administracionelectronica.gob.es/?_nfpb=true&_pageLabel=P80312406127230099567 5&langPae=es.

[137] A forge is a software repository allowing the collaborative development of software over the Internet.

[138] Dirección General para el Impulso de la Administración Electrónica del Ministerio de Hacienda y Administración Pública, *Reutilización de activos y aplicaciones en la Administración*, August 2012, available at http://www.cenatic.es/publicaciones/divulgativas?download=135%3Areutilizacion-de-activos-y-aplicaciones-en-la-administracion.

[139] More information available at http://observatorio.cenatic.es.

results of a survey aiming at assessing the use of FOSS in the Spanish Government in 2010[140]. The findings of the survey can be summarised as follows:

- The majority of the organisms of the national administration (nine out of ten) are using some FOSS. From a quantitative point of view, 40% of server technologies and 15% of desktop technologies are FOSS.
- Outstandingly, 68% of the surveyed organisms acquired FOSS by simply downloading it from a repository or a forge, and 46% of them have developed software based on FOSS (server software).
- One third of the surveyed organisms have procured FOSS (14,7% having required FOSS solutions if possible, and 21,7 % valorising the fact that the proposed solution be FOSS based). However 38,5% have confirmed that they do not adopt any specification in their tenders on that respect.
- 27% of the surveyed organisms confirm having reused FOSS solutions developed by other public administrations.

However, the surveyed administrations also let know that their IT departments needed more personnel specialised in FOSS and that more training was needed. 86% of them deemed necessary to improve the knowledge in FOSS.

This legislative initiative has been confirmed and further extended at national level by the Act 18/2011 regulating the Use of ICT in the Administration of Justice[141], which restates[142] the rules regarding the reuse of software via FOSS licensing as adopted in the eGov law and the Royal Decree.

This general legal framework has also inspired the administrations of the autonomous communities. Indeed, the Basque Country has, in turn, adopted a decree to establish a general principle of openness which is not limited to the eGov applications but applies to any software owned by the public administration[143].

3.3.3 Features

• ACTION:	Legislation
• DECISION LEVEL:	National
• ACTION LEVEL:	Any level
• OBJECTIVES:	To foster reuse of administration software by promoting and explicitly authorising the application of FOSS licences
• MEASURES TAKEN:	Legal authorisation to redistribute software free of charge under a FOSS licence
	Creation of a technology transfer centre listing and hosting the reusable software
	Legal obligation to consider the existing reusable software when procuring software.
• LICENSING:	The EUPL is the “default” licence, but other copyleft licences are valid alternatives

[140] *El Software Libre en los Organismos Públicos de Ámbito Estatal*, Cenatic, December 2011, available at http://www.cenatic.es/publicaciones/onsfa?download=117%3Ael-software-libre-en-los-organismos-publicos-de-ambito-estatal.

[141] Ley 18/2011, de 5 de julio, reguladora del uso de las tecnologías de la información y la comunicación en la Administración de Justicia, available at http://www.boe.es/buscar/doc.php?id=BOE-A-2011-11605.

[142] Article 55 et seq.

[143] Decreto 159/2012, de 24 de julio, por el que se regula la apertura y reutilización de las aplicaciones informáticas de la administración pública de la Comunidad Autónoma de Euskadi, available at http://www.euskadi.net/cgi-bin_k54/ver_c?CMD=VERDOC&BASE=B03A&DOCN=000111019&CONF=/config/k54/bopv_c.cnf

- EFFECTIVENESS: Public Administrations seem globally informed on FOSS.

  Administration software has been reused.

  Positive discrimination has sometimes been adopted.

  There is no clear information about whether the playing field is actively levelled.

3.4. United Kingdom: Government ICT Strategy

3.4.1 General presentation

In March 2011, the United Kingdom's Cabinet Office issued a document officialising the adoption of a new Government ICT Strategy[144]. Cutting costs serves as a leitmotiv[145] and sharing software as a means to an end. The document states that its global aim is openness towards people, the organisations that use its services, and towards any provider regardless of size. The strategy stresses the need to let SME's access the market, to recentre in-house capacities and to avoid oversized, and thus very costly, projects[146]. The government deems it also important to streamline and centralise the procurement practices. To do so, it has planned to develop a new operating model for departments and a new procurement system.

The Government ICT Strategy also aims at fostering the reuse and adaptation of systems which are available 'off the shelf' or have already been procured by another part of the government. Paragraph 15 of the Strategy explicitly states that the different departments will reuse and share ICT solutions and contracts, rather than purchasing new or bespoke solutions and that the government will not commission new solutions where something similar already exists. To identify reusable applications, equipment and resources, the government builds up a cross-government asset register and also plans to create an online Applications Store.

In the same line of reasoning, the government has decided to impose compulsory open standards and to create a level playing field for open source software. Paragraph 16 of the Strategy provides that "*where appropriate, government will procure open source solutions. When used in conjunction with compulsory open standards, open source presents significant opportunities for the design and delivery of interoperable solutions*".

To create the level playing field for the use of innovative ICT solutions, the government has published a toolkit for procurers on best practices to evaluate the use of open source solutions[147], and which encompasses, amongst others, a general document explaining what open source is[148], an open source applications reference list detailing applications that can be contemplated as options for the administration[149] and guidelines on FOSS procurement[150]. The latter only provide high level advice on how to ensure that open source software is fairly considered when procuring an ICT solution. They underline the inherent flexibility of Open Source as regards several requirements that should always be considered when procuring software, such as the scalability of licence, the transferability of software or the compliance with open standards.

The guidelines also explain that *"where the software is free to use 'gratis' software and all associated products are free for the whole of life use then there is no requirement to tender the requirement for the licenses"*. However, "*(a) purchase of support and maintenance procured separately from licenses will need to be tendered where it is expected that the cost of support meets the EU thresholds and in accordance with any standing financial instructions".*

[144] Available at http://www.cabinetoffice.gov.uk/content/government-ict-strategy
[145] This is confirmed by Linda Humphries on the Government's blog, "Are open standards a closed barrier?", 12 avril 2012, available at http://digital.cabinetoffice.gov.uk/2012/04/12/are-open-standards-a-closed-barrier/
[146] The Government sets a presumption against government ICT projects valued at over £100 million.
[147] Available at https://www.gov.uk/government/publications/open-source-procurement-toolkit.
[148] "All about open source: an introduction to Open Source software for Government IT", version 2, April 2012, available at the same address.
[149] "Open Source Software Options for Government", version 2, April 2012, available at the same address.
[150] "ICT Advice Note - Procurement of Open Source", October 2011, available at the same address.

The Strategy also includes the establishment of an Open Source Implementation Group, a System Integrator Forum[151] and an Open Source Advisory Panel[152] to assist with the deployment of agile[153] solutions using open source technology and to educate, promote and facilitate the technical and cultural change needed to increase the use of open source across the government. It also envisages the creation of a 'virtual' centre of excellence across the government and the private sector which can enable fast start-up and mobilisation for such agile projects.

3.4.2 Results

Open source advocates such as the Open Forum Europe welcomed the UK Government's "*determination to move the public sector in the UK away from being locked in to large scale single supplier proprietary software solutions*"[154]. Criticising the reluctance that governments showed until then to consider open alternatives, the Open Forum Europe expressed its yearning to observe concrete results: "*it is in procurement that the Strategy will either come alive or wither."*

In its report[155] of May 2012, the government sets out the progresses achieved over one year of implementation. As far as open source is concerned, the report only refers to the publication of the procurement toolkit and confirms the establishment of the Open Source Advisory Panel. An e-petition site, which has been built in 8 weeks on open source software and using open standards, is reported as a success story. No other figure or example is provided.

In its assessment of June 2012, the Institute for Government (an independent charity helping to improve government effectiveness) does not report much more on concrete results in open source adoption[156]. On the contrary, it stresses the ICT leaders' view that the focus should be on enabling the government to perform more effectively and not on implementing the ICT strategy "in a tick box fashion".

The press reported that the Strategy is largely lobbied against[157], and that during a round table event organised by the Cabinet Office, open standards opponents easily dominated a meeting motion against the government's open standards policy[158]. The definition of the "open standard" concept is the crux of the tension. Reporting on the event, a representative of the government observed that *"the consensus was that the definition and proposed policy would be detrimental to competition and innovation"* [159].

This battle around open standards questions the sustainability of the Strategy in general and seems therefore to also have a detrimental effect on open source adoption.

3.4.3 Features

- ACTION: Policy (Governmental Strategy)
- DECISION LEVEL: National
- ACTION LEVEL: National

---

[151] See http://www.computerweekly.com/blogs/public-sector/2011/02/24/open-source-si-forum.pdf.

[152] A list of the membres of the Open Source Avisory Panel has been communicated in the framework of a parliamentary question, available at http://www.publications.parliament.uk/pa/cm201011/cmselect/cmpubadm/writev/goodgovit/it65.htm.

[153] Agile software development is a software development method where requirements and solutions evolve flexibly through collaboration between self-organizing, cross-functional teams.

[154]See the OFE's press release of 30 March 2011, http://www.openforumeurope.org/press-room/press-releases/openforum-europe-welcomes-the-publication-of-the-uk-governments-ict-strategy.

[155] "One year on : Implementing the Government ICT Strategy", May 2012, available at https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/61950/One-Year-On-ICT-Strategy-Progress.pdf.

[156] "System upgrade? The first year of the Government's ICT strategy Features", June 2012, available at http://www.instituteforgovernment.org.uk/sites/default/files/publications/System%20Upgrade.pdf.

[157] G. MOODY, « UK Government Open Standards : the great betrayal of 2012 », Computer World UK, 22 December 2011, available at http://blogs.computerworlduk.com/open-enterprise/2011/12/uk-government-open-standards-the-great-betrayal-of-2012/index.htm.

[158] "Proprietary lobby triumphs in first open standards showdown", Computer Weekly, 13 April 2012, available at http://www.computerweekly.com/cgi-bin/mt-search.cgi?blog_id=102&tag=BCS%20Open%20Source%20Speclialist%20Group&limit=20.

[159] "Are open standards a closed barrier?", *op. cit.*

• OBJECTIVES:	To foster the reuse of software across the administration. To level the playing field for open source solutions.
• MEASURES TAKEN:	Publication of a toolkit (set of guides) Setting up expert panels and forums Maintaining an asset register and an applications store Operating a ‘virtual’ centre of excellence across government
• LICENSING:	Not specified (“open source” is the term used).
• EFFECTIVENESS:	There does not seem to be any important achievement so far, but it seems also too early to draw conclusions. Lobbies are actively opposing the adopted open standard strategy, and the government seems open to reconsidering its position. This situation also negatively impacts FOSS adoption.

3.5. Belgium: IMIO (inter-municipal company)

3.5.1 General presentation

IMIO (*Intercommunale de Mutualisation Informatique et Organisationnelle*) is a government owned inter-municipal company that has been incorporated on 28 November 2011 (under the form of a limited liability cooperative company) by a partnership of ten Walloon municipalities with the blessing and support of the Walloon Region[160], which is the supervisory and approving authority[161] of the Walloon municipalities.

IMIO has not been created from scratch, as it is based on the previous “CommunesPlone” project[162], a collaborative “bottom up” approach which gathered many Walloon municipalities aiming at gaining independence from IT services providers by developing, essentially by themselves and in a cooperative manner, applications and websites for their own use as well as for their citizens. The CommunesPlone community was composed to a large extent of IT workers employed by the municipalities involved or by the SME’s providing the services to the latter and to the public company. IMIO has taken over the CommunesPlone project and pushed it further by providing an official, logistical and incorporated structure.

The statute of the company provides that its statutory objectives are to promote and foster the mutualising of organisational solutions and of IT products and services for the local authorities of Wallonia. To do so, IMIO must either act as a central procurement agency which will procure software via public tenders, or develop internally software applications which are mutualised and distributed under a free licence. In the latter case, IMIO is expected to manage a free software patrimony which must be coherent and robust and which belongs to the public administrations. IMIO must ensure that it has internally the technical control of the software, and that the latter will evolve, remain sustainable and be distributed in compliance with the applicable free licence. The statutes further specify that the company produces and mutualises, amongst others, Plone-based open source software (Plone being a Content Management System licensed under the GPLv2).

The three main activities of IMIO are[163]:

- Producing (procuring, developing or procuring the development of) open source software to meet the needs of local authorities (IMIO works also with a network of open source SMEs)[164];

[160] In its Regional Policy Declaration of 2009-2014, the Walloon government has set as one of its objective to promote the use of free software. See “Projet de Déclaration de politique communautaire 2009-2014”, available at http://www.awt.be/contenu/tel/awt/declaration_politique_regionale_2009_2014.pdf.
[161] “Autorité de tutelle”.
[162] See the IDABC study « Networks effects : Plone for Belgium and Beyond », available at https://joinup.ec.europa.eu/elibrary/case/networks-effects-plone-belgium-and-beyond-0.
[163] More information is available at http://www.imio.be/presentation.
[164] See the Joinup news “Walloon communities sharing software as an alternative to procurement” available at http://joinup.ec.europa.eu/news/walloon-communities-sharing-software-alternative-procurement.

- Buying proprietary solutions in purchasing centres to provide solutions at lower costs and a support service; and
- Formalising work processes for the local government.

IMIO provides its software and services to any municipality of the Walloon region, which can become member of the project. Currently, it is partly financed by subsidies of the Walloon region, and partly by the prices paid by the members for each “product”. These prices are linked to the number of inhabitants of the municipality. The solutions proposed by IMIO can be deployed on the infrastructures of their members or made available in "SaaS Software as a Service" from IMIO’s infrastructure.

3.5.2 Results

IMIO has currently more than 180 members, amongst which more than half of the Walloon municipalities (150 out of 262 – the 30 remaining members being other types of public services such as public social action centres).

IMIO benefits from Walloon Region subsidies (1.2 Million Euro at its launch), but has already reached a turnover of about 1 Million Euro in 2012. It is expected to become fully sustainable within a few years.

According to Joël Lambilotte (co-founder of CommunesPlone and employee of IMIO) “*the IMIO initiative is successful and the results are better than what was predicted in the business plan. The success comes from the “official” status of the organisation as a publicly owned entity. A second success factor is certainly the experience gathered from the field with the CommunesPlone project, whose debut dates back to 2005. The partnership with many technological SME’s has also provided an undeniable advantage comparing to other solutions available on the Market. These elements bring a very hands-on and bottom-up approach to the global strategy*”.

3.5.3 Features

• ACTION:	Policy (joint action of several local PA)
• DECISION LEVEL:	Local (Walloon region & municipalities)
• ACTION LEVEL:	Local (Walloon municipalities)
• OBJECTIVE:	To foster the mutualising of local administration software
•	MEASURES TAKEN: Creation of a public company which procures, develops, maintains, supports and mutualises software for the local municipalities.
• LICENSING:	Open source software (GPL is mainly involved)
• EFFECTIVENESS:	The initiative seems successful and reaching its objectives. It is too early to assess its sustainability.

3.6. France: Circular on the use of open source in the administration

3.6.1 General presentation

On 19 September 2012, the French Prime Minister Jean-Marc Ayrault addressed a circular to all the French ministers inviting them to implement the guidelines on the use of free software in the administration[165] prepared by the DISIC (*Direction Interministérielle des Systèmes d’Information et de Communication*).

The guidelines start from the statement that “*from now on, in order to meet business needs, Free Software must be considered on equal footing with other solutions*”. After some introductory explanations on the basic features of FOSS and its licensing scheme, its model based on services and its advantages in different contexts, the guidelines broadly describe

[165] “Usage du logiciel libre dans l’administration”, annexed to the circulaire 56/SG of 19 september 2012, available at http://circulaire.legifrance.gouv.fr/pdf/2012/09/cir_35837.pdf. An english translation made by the APRIL is available at http://www.april.org/en/french-prime-minister-instructions-usage-free-software-french-administration.

eight inter-ministerial action lines aiming at facilitating the use of Free Software solutions in the administration's choices and at levelling the playing field, while at the same time achieving maximum economic efficiency and quality:

- Instituting an effective convergence on certain Free Software projects. A convergence framework is established, which aims at selecting and focussing some relevant FOSS that can be developed and reused in State information systems. Each ministry must participate in its updating and progressive reinforcing.
- Activating networks of experts, which gather specialists from the different ministries and aim at sharing expertises and competences. This includes the organisation of inter-ministerial workgroups and public workshops and conferences.
- Improving Free Software support in a controlled economic context. The objective is to centralise and to mutualise support and maintenance services amongst the ministries.
- Contributing in a coordinated way to chosen free software projects. The government plans to financially endorse Free Software development by systematically re-injecting from 5 to 10 percent of the avoided licensing costs in the development process.
- Keeping in contact with the large communities. "*Just as software editors maintain regular contact with all ministries, to update knowledge of their products, be able to anticipate their changes, and even assess needs, it is essential to have links to large communities such as the Mozilla Foundation or the Document Foundation. However, as these foundations do not have a commercial approach, the logic is reversed. It is the administration that must regularly contact them*", the document explains.
- Deploying credible and operational alternatives to the large software editors' solutions. The aim is to identify and focus on credible alternatives and foster their adoption.
- Mapping out use of FOSS and its impacts. Annual analyses on FOSS adoption should be carried out and published.
- Developing a culture of use of Free Software licenses in the development of public information systems. This last point acknowledges and aims at addressing the "complex management of code ownership". It provides that "*the State must safeguard its ability to release code in a manner that maximizes its own benefit, regardless of which provider did the development. The State must therefore make use, or prepare the use, of Free Software licenses, be they permissive or not, depending on the context. It must also ensure that this freedom prevails vis-à-vis its suppliers in every context that could lead to reuse, unless explicit additional costs are generated*".

  To achieve all these results, some concrete action lines are planed:

  - a network of experts is established among lawyerss/purchasers involved in the drafting of specifications and administrative clauses;
  - specific training courses are set up within ministries: fast-track ones for project managers and developers, more in-depth ones for lawyers and buyers;
  - provider liability clauses and obligations must also be added when said providers use or develop Free Software code, and
  - licence management must be one of the components of the explicit IT governance within each ministry.

It is noticeable that, in order to add legitimacy to the guidelines, reference is made to the Council of State's decision of 30 September 2011[166], which confirmed that a public

[166] Conseil d'Etat, Decision n°350431 of 30 September 2011, available at http://arianeinternet.conseil-etat.fr/arianeinternet/getdoc.asp?id=192208&fonds=DCE&item=1.

administration can freely select a Free Software which is inherently « freely accessible, free of charge and modifiable by any service provider » and procure customisation, installation and maintenance services in relation to that particular software.

3.6.2 Results

Benjamin Jean (free software specialist and advocate)[167] welcomes and appreciates the relevance and clear-sightedness of the circular, but regrets the absence of the local administrations, which could have also been involved in the sharing and mutualising process.

The circular has been welcomed by the APRIL (French association for the promotion and defence of free software) as good news, but the association underlines that this decision from the French State is just a first step which needs to be further implemented. The association notices that the document provides only high level guidelines that must be further detailed and implemented by taking many concrete measures[168].

3.6.3 Features

• ACTION:	Policy (ministerial circular)
• DECISION LEVEL:	National
• ACTION LEVEL:	National
• OBJECTIVES:	Levelling the playing field
	Fostering the use and mutualisation of free software
• MEASURES TAKEN:	Selection of a set of credible free software alternatives
	Creation of expert networks
	Free software monitoring
	Contributing to Free Software development
	Developing a culture of FOSS use
• LICENSING:	Reference to “free software”
• EFFECTIVENESS:	It is too early to assess the concrete results of the initiative, which consist of high level guidelines.
	The initiative requires an important implementation work that remains to be determined and carried out.

4. OBSERVATIONS

What first strikes the observer when comparing the different cases described in this briefing paper is the diversity of the initiatives. Whereas the logic lying behind them is usually evolving around the same concerns and objectives, the adopted strategies and concrete actions are very diverse in terms of scope, scale, means and ambitions. Furthermore, they are not at the same stage of development and implementation. This, in addition to the cultural and state structure differences, renders any meaningful comparison difficult.

All the initiatives aim at improving the public procurement practices and stem from the observation that even though FOSS presents inherent characteristics that correspond to good ICT governance principles, the option is not considered enough when choices are

[167] B. JEAN, « Synthèse sur la publication par le Premier Ministre Jean-Marc Ayrault de la circulaire du 19 septembre 2012 présentant des orientations et des recommandations sur le bon usage des logiciels libres dans l'administration française », 27 septembre 2012, available at http://blog.vvlibri.org/index.php?post/2012/09/27/Synth%C3%A8se-sur-la-publication-par-le-Premier-Ministre-Jean-Marc-Ayrault-de-la-circulaire-du-19-septembre-2012-pr%C3%A9sentant-des-orientations-et-des-recommandations-sur-le-bon-usage-des-logiciels-libres-dans-l-administration-fran%C3%A7aise.

[168] « Analyse détaillée de la circulaire Ayrault sur le bon usage des logiciels libres dans les administrations », APRIL, 8 novembre 2012, available at http://www.april.org/analyse-detaillee-circulaire-ayrault-sur-le-bon-usage-des-logiciels-libres-dans-les-administrations.

made. Raising awareness seems to be the first (explicit or implicit) objective, and effect, in all cases.

The guidelines attached to the French Ayrault Circular are particularly interesting on that aspect as they stress that one of the main causes of the lack of awareness of FOSS is the fact that, contrary to proprietary software, FOSS is usually not the subject of ongoing marketing and promotion practices. Therefore, FOSS and their communities should indeed be actively monitored by the administrations' procurement officers, and preferably by dedicated open source experts. The importance of the active participation of the administration IT staff in the community is also illustrated by the Belgian IMIO project, which has been created on the top of a community of developers (CommunesPlone) composed to a large extent of IT workers employed by the municipalities involved or by the SME's providing the services to the latter and to the public company. IMIO is therefore entirely integrated in the community and is one of its main actors.

Part of IMIO's success is also due to its bottom-up organisation, which embraces the "traditional" open source ways[169]. It has been created by the municipalities for the municipalities, on the basis of a general observation: on the one hand, each one of them disposes of very limited budgets and resources to procure or develop specific management tools, CMS, websites, e-Gov platforms; on the other hand, they all share the same needs. The municipalities realised that pooling resources to develop a pool of common software was therefore the natural way to address the issue.

This approach is very different from the NOIV program, the French Circular, the UK strategy or the legislative approaches, which are typical of top down governance. In such type of initiatives, any factor of resistance to change must be carefully analysed and integrated in the strategy. A good strategy should include an awareness phase and, according to Paapst, a subsequent persuasion phase with four dimensions: a legal dimension, a technical dimension, a financial dimension and a subjective "knowledge/experience" dimension. Within this subsequent phase different elements influence the degree of willingness to adopt and use a new strategic IT policy in any of the four identified dimensions. According to Paapst, a reason why the NOIV has not been as successful as expected is that "*for instance the mere use of the legal instrument (e.g. the European procurement guidelines) is not enough to change behaviour and to counterbalance negative influences coming from within the technical dimension and the experience/knowledge dimension*"[170]. By welcoming the Italian initiative with caution, Piana & Aliprandi confirm Paapst's theory: purely financial reasons are equally not enough to ensure a successful migration to FOSS[171].

Once a public administration is aware and convinced that FOSS is good for its ICT, it can draw many teachings from the experiences analysed in this briefing paper.

Even before considering a call for tender, there is a consensus amongst the authorities involved that downloading FOSS free of any charge or compulsory fee can be a valid means of acquiring software without the requirement of a competitive bidding[172]. Once the FOSS is selected and downloaded, paid services and support for such software can be acquired via the traditional public contract process. Such method has been validated by the French Council of State.

In the framework of a call for tenders, the Italian Constitutional Court teaches that the concept of FOSS is independent of any given technology, brand or product but refers to a contractual regime that can be preferred without damaging competition. References to the concept of free and open source software are therefore always legal (contrary to the use of

[169] E. RAYMOND's "The Cathedral and the Bazaar" describes the bottom-up software design approach of the Linux community. It is available at http://www.catb.org/esr/writings/homesteading/.

[170] M. PAAPST, *Barrières en doorwerking : Een onderzoek naar de invloed van het open source en open standaarden beleid op de Nederlandse aanbestedingspraktijk*, PhD thesis defended on 10 January 2013, available at http://irs.ub.rug.nl/ppn/353037710.

[171] *Idem.*

[172] This is also confirmed in the European Commission's IDABC programme's "Guideline on public procurement of Open Source Software" of March 2010, available at http://joinup.ec.europa.eu/sites/default/files/OSS-procurement-guideline%20-final.pdf.

trademarks or specific technologies that should be, as a general rule, banned).

The NOIV's examples of award criteria are a good source of inspiration, as they make use of terms, concepts and objectives that are as neutral as possible. In contrast, it is interesting to note that the explicit reference to the EUPL by the Spanish interoperability framework has been source of discomfort for FOSS-based IT providers, given that a vast majority of open source applications are available under other licences (mainly of the GPL family) that do not allow relicensing under EUPL. Fortunately, the law explicitly allows the use of other licences, and one must hope that Spanish administrations carefully and wisely assess the necessity to specifically require the EUPL[173].

The current public procurement regulatory framework seems therefore not to constitute, as such, a hindrance to the adoption of FOSS by administrations. It provides ways to develop practices that aim at levelling the playing field or preferring the procurement of FOSS, if there is a will to go in that direction. The analysed cases illustrate that this last condition is probably the one that requires the most attention: whereas the policy shapers are aware of the advantages of FOSS, policy takers show different degrees of resistance, which is motivated by multiple factors that must be duly analysed and taken into consideration, and that are sometimes overlooked.

Passing laws could be contemplated as a means to override the resistance effect thanks to the compulsory nature of the instrument used. The Spanish and Italian experiences illustrate, however, that such exercise is complex, as the adopted law is likely to interfere with copyright, competition or procurement laws and principles. The law must therefore be cautiously drafted and should not damage competition nor result in a technological stagnation.

The Spanish law is quite astonishing as it is drafted in a way that it *allows* administrations to share software using FOSS licences. Besides the symbolic aspect of this explicit authorisation, one would tend to wonder what concrete change is brought by such law to the general regulatory framework: would such FOSS licensing practice not have been legal anyway without such positive statement? Furthermore, a devil's advocate would even argue that the Spanish law restricts FOSS licensing practices in administrations as it seems to impose the use of copyleft licences. On the one hand, such requirement restricts the spectrum of possible scenarios (as copyleft can generate compatibility problems in heterogeneously licensed developments)[174], whereas on the other hand, it implies the use of licences that must be handled with more care (as copyleft usually entails more obligations to comply with). In contrast, the Italian law considers FOSS as a self-justifying criterion (whereas the choice of proprietary solution must be specifically explained) but it does not further require a specific type of FOSS licence.

[173] On this regards, see the ISA programme's "Standard "sharing and re-using" clauses for contracts", https://joinup.ec.europa.eu/sites/default/files/ISA_Share_Reuse_D_2%201%20Standard%20Sharing%20and%20re-using%20clauses%20for%20contracts_final%20version.pdf.

[174] PH. LAURENT, "Free and Open Source Software Licensing: A reference for the reconstruction of "virtual commons?", *op. cit.*

ANNEX: COMPARISON TABLE

	Dutch NOIV	**Piedmont Region's Act**	**Spanish NIF**	**UK Government ICT Strategy**	**Walloon IMIO**	**French Ayrault Circular**
Action	Policy	Legislation	Legislation	Policy	Policy	Policy
Decision level	National	Local	National	National	Local	National
Action level	Any level	Local	Any level	National	Local	National
Objectives	Awareness Level playing field Preference	Preference	Reuse of software	Reuse of software Level playing field	Software mutualisation	Use and mutualisation of FOSS Level playing field
Measures Taken	Promotion Support office Guidance & Support Guidelines on award criteria	Law establishing procurement rules	Authorisation to use FOSS licences Obligation to reuse Technology transfer centre	Toolkit (guidelines) Expert panels Asset registers & app. store Centre of excellence	Creation of an inter-municipal public company	Selection of credible free software alternatives Expert networks Free software monitoring Contribution to FOSS development "Culture" of FOSS use
Licensing	Open Source EUPL considered	Free Software	EUPL (default licence) Other copyleft licences	Open Source	Open Source GPL mainly involved	Free Software
Effectiveness	Objectives not reached	Law in application Court validation	Awareness Reuse Some positive discrimination	The strategy is lobbied against Too early to draw conclusions	Objectives reached so far	Too early to draw conclusions

Philippe Laurent is Senior Researcher at the CRIDS (Research Centre - Information, Law and Society of the University of Namur) and Lawyer at the Brussels Bar (Marx Van Ranst Vermeersh & Partners). As a researcher, Philippe mainly studies intellectual property licensing, data and software protection, copyright limitations, open source and open content schemes, cloud computing and the governance of the Internet. He wrote several reports and articles on open source licensing and is currently working on the development of a local FOSS expertise centre. Philippe's work as attorney-at-law focuses on intellectual property & IT law, data protection, trade practices, distribution agreements, advertising, as well as on broader commercial law matters. Philippe is also appointed by the CEPANI as Third-Party Decider for ".be" domain name disputes and is alternate member of the copyrights and neighboring rights section of the Intellectual Property Council of the Belgian Ministry of Economy.

Legal aspects of free and open source software in procurement: the example of the City of Munich

Oliver Altehage, Kirsten Böge & Dr. Jutta Kreyss

CONTENT

1 ORIGIN AND ORGANISATION OF THE LIMUX PROJECT

Bavaria's capital, Munich, recently finished an IT migration project regarding its clients. In a project like this, the focus is often on the technical aspect, while the cultural and human dimensions are neglected. Yet an IT project on such a large scale requires a holistic approach, taking into account the organisational preconditions, political factors, individual habits and the cultural specificities of a large public-service organisation (Theuvsen et al. 2010). A holistic approach is built around comprehensive stakeholder management, which means strategic planning of relations with the project's main stakeholders. It is thanks to shrewd cooperation with the main protagonists, such as the City Council, Staff Committee, management and also the open source community, that in 2012, a good nine years after the LiMux project began, and despite all the imponderables, the project was successfully finished. It is thanks to its solid relations with the principal stakeholders that the project – often referred to as a 'flagship' project[175] – has the requisite stability.

Before a full description of all the stakeholders, it seems helpful to provide a few key data about the project.

The aim of the LiMux project was to migrate all the approximately 15 000 PC work stations present in 11 business units and 4 municipal undertakings to an open source-based, standardised and consolidated system. To put it precisely, all the PC work stations used by the administration of the City of Munich are to be equipped with Open Source office systems and at least 80% of all PCs are to run on a Linux-based operating system.

The LiMux project was intended to progressively **eliminate the legacy of dependence** on proprietary products and in the long term attain the desired **flexibility** of software and architecture. As a rule, although using products tailored to one another which are available from a single producer is convenient in the sense that functions can be used in common or (proprietary) file formats can be used throughout the organisation, there are also

175 **'Flagship project'** is a term used to describe an exemplary project which, in addition to fulfilling its particular purpose, is also intended to set an example which can be followed by numerous subsequent projects. Apart from success, therefore, the aim is to ensure that the project is widely known. (http://de.wikipedia.org/wiki/Leuchtturmprojekt, 16.11.2011)

drawbacks because this makes it significantly more difficult to replace these products, the easiest option then being to purchase further products from the same manufacturer. This gives rise to costs and dependences which could have been avoided. Ultimately, this significantly restricts freedom of choice of appropriate IT systems within an organisation. Moreover, the LiMux project is seen as a blueprint for the project culture approach to be adopted in future IT projects.

The approach adopted by Munich City Council in 2003 was based on three fundamental decisions:

1. to introduce a free, open-source operating system, including office communications based on open standards for all work stations;
2. that in the future all specific administrative procedures should be acquired or developed as being platform independent;
3. that a standardised IT platform with consolidated applications and databases should be used. When the project began, there were 21 IT business units, more than 1000 applications, some of which were redundant, innumerable versions, no uniform template system and, apart from a central LDAP - Lightweight Directory Access Protocol - server, absolutely no standardisation throughout the city.

In the early summer of 2005, the work station migration project was then launched.

As of December 2011, all staff was working on their PCs with the free office communication products OpenOffice.org, Firefox and Thunderbird, and more than 9 000 work stations have migrated to the Linux-based operating system.

This makes Munich the biggest public-sector open-source project in Germany with high visibility. This would already be a sufficient reason to write about it. But the aim of this note is also to describe it from a particular point of view, with reference to the significance of important stakeholders in the project. What contribution are politicians, management, staff or the open source community making to the success of the project?

2. STAKEHOLDER MANAGEMENT IN THE PUBLIC SERVICE

The term ‘stakeholder’[176] is derived from the word ‘stake’, which can mean an asset risked in gambling or an investment or interest in a business. ‘Holder’ meanwhile refers to the person who owns or possesses it. ‘Stakeholder group’, or simply ‘stakeholders’, is a good way of referring to the main parties with an interest in the LiMux project.

With the focus on the relevant stakeholder groups, the desire arises for successful ‘relations management’. How can one communicate with the main stakeholder groups in a manner which they appreciate, and how can those relations be successfully maintained?

According to the definition provided by ISO 10006 (http://www.iso.org), stakeholders in a project are all those who have an interest in the project or are affected by it in any way.

A distinction is made between *active* and *passive* stakeholders (Freeman 1984). Active stakeholders are directly involved in working on the project (e.g. team members) or are directly affected by it (e.g. users, suppliers, business management).

Passive stakeholders are only indirectly affected by the implementation of the project or by its effects (e.g. representatives of interest groups, associations, etc.).

Moreover, each project has its specific stakeholders, whose significance needs to be established by means of a force field analysis (Lang 2010). In such an analysis, various interests and attitudes of relevant groups can be analysed to create a portfolio:

176 For the concept ‘stakeholder’, see http://wirtschaftslexikon.gabler.de/Definition/anspruchsgruppen.html

- influence on the project (power and influence from above)
- attitude towards the project (commitment to the aims of the project).

In order for a project to be successful, it is therefore important to know the relationship between the individual stakeholders and the project.

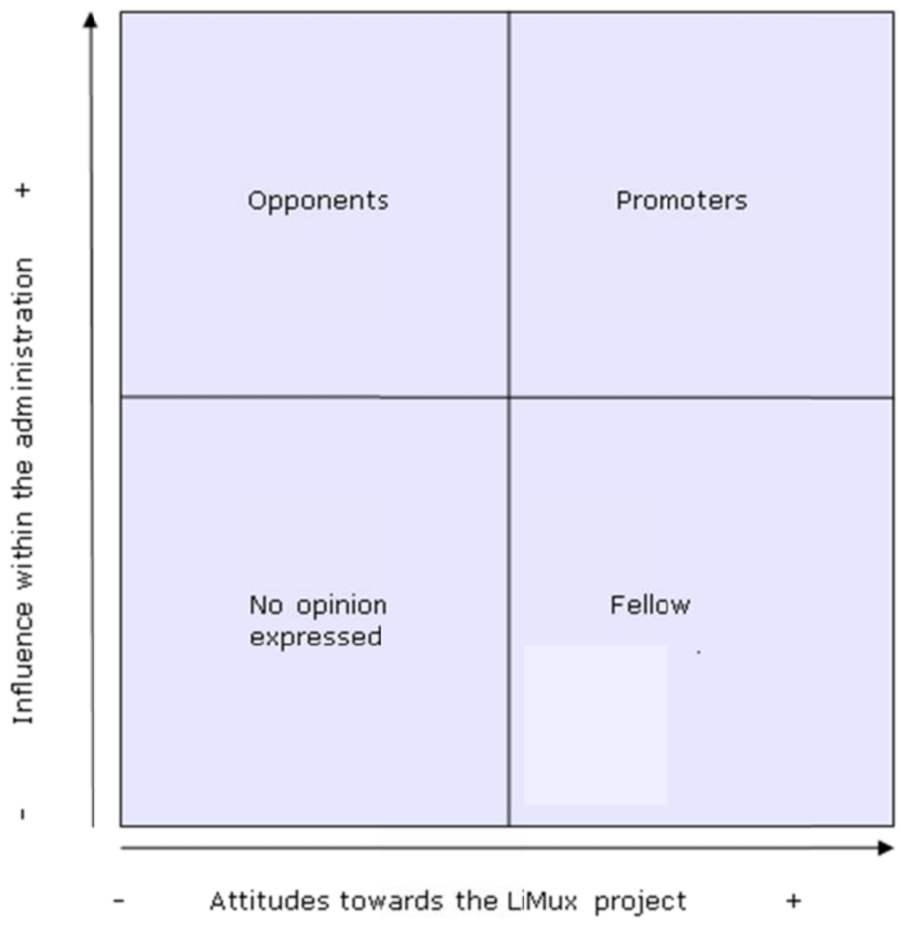
The outcome of the stakeholder analysis is the basis for planning regular communication, developing a role-based structural model, the communication plan and much else besides.

How exactly should the stakeholder groups be systematised and positioned?

The Y axis indicates how influential a stakeholder group is in the capital of Bavaria, Munich. In other words, how many other people hear and act on its recommendations/ decisions/instructions?

The X axis clarifies what attitude the same stakeholder group has towards the LiMux project. I.e.: how valuable does it consider the performance of the project to be? How strongly does it support the aims of the project?

Figure: Matrix of the analysis of the lines of forces


Influence within the administration
+
-
Opponents
Promoters
No opinion expressed
Fellow
-
Attitudes towards the LiMux project
+

From the combination of the respective axes, a figure can be drawn which identifies four characteristic profiles: those who express no opinion, fellow campaigners, opponents and promoters.

- Stakeholders who express no opinion: they do not commit themselves either way with regard to the aims of the project, and have little influence within the city administration.

- Fellow campaigners: these are people who are strongly committed to the aims of the project, but at the same time have limited influence over other stakeholder groups within the city administration.
- Opponents: these are influential stakeholders who actively oppose the aims of the project.
- Promoters: a group who is wholeheartedly working to make the project a success, even in the face of resistance within the city administration.

At the beginning of virtually any major project affecting a whole organisation, the various stakeholder groups are in a ‘neutral’ zone vis-à-vis the aims of the project. Only in the course of the first few months do the stakeholder groups decide what position to adopt. The aim of successful stakeholder management is to turn the stakeholder groups into protagonists bearing a shared responsibility. The extent to which the core team and the principal stakeholders of the LiMux project in Munich have managed to achieve this will be indicated below.

3 STAKEHOLDERS IN THE LIMUX PROJECT IN THE CAPITAL OF BAVARIA, MUNICH

The following list of stakeholders is based on the history and administrative structure of Munich. It is not definitive, nor could it be, but rather only contains a representative sample, looking at the situation from a temporally defined angle. All the stakeholders listed are described with reference to the measures and finally positioned in a force field portfolio.

3.1 The project team: the starting point for all stakeholder groups

The LiMux project team comprises a core team and an extended project team. The core team consists of approximately 25 people who are working on the development and provisioning of the LiMux Client, Support for the Open Office Components such as OpenOffice, Thunderbird and Firefox, including the conversion of forms and macros, as well as the further development of, and support for, the WollMux (document and template system[177]), with support from external service-providers. The core team is organised into smaller groups dealing with or identified as:

- Management of requirements,
- Development of the LiMux client (excluding Office),
- Development of the office and WollMux components,
- Migration support and incident management,
- Test management,
- Release management and architecture, and
- Change & communication combined.

The core team is managed by the project management and the project office and the technical lead.

The extended project team consists of numerous colleagues from the migration fields, who decide about the requirements in their respective fields, report on the migration and provide day-to-day support for users. This extended project team is regarded as an independent stakeholder group in the description of the project (IT managers and IT staff).

It took nearly three years to put together the LiMux core team, and in the course of the project the form taken by the team has changed, just as in the past three years the organisational parameters have changed radically. In parallel with the migration of work

[177] WollMux was developed, in the framework of the LiMux project, as an extension of OpenOffice.org and is now being used in more and more municipalities to work with templates, forms and letterheads (www.wollmux.org).

stations to a LiMux Client peculiar to the city administration, a large part of the IT of the city administration has been transferred to a municipal undertaking, IT@M, with took effect from 1.1.2012. And it is precisely for the reorganised Munich IT that the LiMux project is to provide a standardised, open source-based modern IT platform.

One guarantee of success has been the continuity of staffing in the ramp up and migration phase. There have been particularly few changes among the technical staff in these phases. That has been very important, because as a result of the migration to open source-based work stations, the organisation's own responsibility for developing and operating the system is constantly increasing.

The core team is therefore the nucleus of the overall project, i.e. all its strategic and operational decisions have a direct impact on the success of the project. For that reason, we regard it as the prime mover of the project and the origin of its success. Many members of the core team came to the city specifically to work on the project, and, even if they do not regard themselves as open-source evangelists, they do see themselves as promoters of the open source community.

3.2 The City Council: the legal and legitimating authority

The LiMux project initially made it at least into one or two headlines. USA Today even reported the decision by Munich City Council on its front page[178]. Thus, from the outset, the project had a high profile in the media. The Social Democrats and Greens cited Linux in their municipal election campaigns with reference to the concept of freedom, which worked well as advertising. In 2003, 2004 and 2007, the City Council took the main decisions in favour of the project, in some cases with cross-party support; additional decisions have been necessary because the project has been kicked of in 2003, a detailed concept has been finished in 2004 and the budget had to be modified in 2010. In its 2004 decision, the City Council even approved project funding in excess of what had been requested, and expanded the aims of the project. In accordance with the importance thus attached to it, the Deputy Mayor was immediately assigned organisational responsibility for the project.

The controlling political majority is both a fellow campaigner for the project and its sponsor. Even if occasional criticisms of the open source strategy are voiced, the City Council is standing by its decision and bolstering that strategy.

The governance structure is one of the reliable factors in the project. The guidelines for the project are brought into line with the guiding principles of the policy pursued by the controlling majority. This gives the project the requisite legal basis and at the same time imparts political legitimacy to its aspiration to act as a 'beacon of independence'. This strong backbone is needed for a project so sensitive to the type of media cover it receives (see Section 3.10 on the public as a stakeholder).

3.3 An open and supportive Staff Committee

Under the Bavarian Staff Representation Act, the Public Service Staff Committee possesses extensive responsibilities, rights and obligations. It is the dialogue partner for staff and management with regard to such topics as the staff establishment and promotions, equipment in the workplace, cooperation with staff and staff satisfaction. Whenever major changes are planned in the workplace, it is necessary to consult the Staff Committee at an early stage, which is required not only by law but also in the interest of cultural fairness.

The Staff Committee was invited to participate at an early stage, even while decisions were being prepared overall and preliminary studies drafted. The business unit Staff Committee and the overall Staff Committee were kept informed from the moment when the idea was first mooted until the migration was planned, and were also invited to deliver opinions. Project aims and procedures were presented and debated in all decision-making bodies of the Staff Committee. The initial city-wide introduction of an e-learning system was discussed particularly intensively, resulting in an agreement with the Umbrella Staff Committee. Subsequently, the 'LiMux World of Learning' was devised, which in 2007

178 Byron Acohido, USA TODAY, 13.07.2003: http://www.usatoday.com/money/industries/technology/2003-07-13-microsoft-linux-munich_x.htm

received the Eurelia Award for an innovative and employee-friendly IT platform. It supplements the numerous training measures accompanying the migration and the new workplace software solutions.

3.4 The IT high level management team: a bastion of strength in times of disunity

Leadership is what matters – or at least, that is the conventional wisdom. In the LiMux project, it is the IT high level management team that is particularly vital. The team is a bastion of strength in the project. Derived from the above-mentioned governance model, the whole management and leadership team supports the aims of the project. From the Deputy Mayor, via the Director and IT Supervisor of the City of Munich, the Divisional Head within the Directorate and works manager of the municipal undertaking IT@M, and an Advisory Board, to the project leader, everybody is convinced that the project is feasible.

The establishment of regular communication, however, involved a few learning curves. Up to reaching the project's objectives, bimonthly steering committees, monthly reporting deadlines with the IT Supervisor and weekly meetings with the immediate management ensured that, within the administration, the necessary attention was paid to developments in accordance with the aims. If problems arise, a mentor provides the necessary support. The involvement of the business units heads at the highest level is arranged by means of needs-based reports at the weekly reporting sessions by the Deputy Mayor and/or the IT Supervisor of the City of Munich.

The perseverance of the management hierarchy secured the project at the times of greatest peril.

The IT management team is the promoter of the project, and always keeps an eye on the practical constraints of the business units and municipal undertakings.

3.5 The role of IT managers as driving forces and opinion formers

Such a far-reaching project – affecting every single work station – requires excellent cooperation between the core teams and the migration fields. It is at this interface that the **unit project leaders** within the business units are located. Much effort was required in order to define specifically the roles of the unit project leaders in the respective migration fields and then for the relevant business unit heads to appoint staff to these posts. From the outset, this stakeholder group gave considerable impetus to the project in the form of critical, but always constructive, observations. Here, as in the case of the technical managers (see below), it was necessary to drive cultural change by introducing an open source strategy in the face of some resistance. The role of the unit project leaders is central to the success of the project, because they bear operational responsibility for the migration in their own fields. Today, the unit project leaders are at least fellow campaigners for the new IT Strategy, if not indeed promoters of it, even if they always continue to criticise, because their whole objective is the smooth provision of services by the City of Munich to its citizens.

This change of attitude has been assisted by a series of measures:

- An **initial strategy** at the beginning of the project helped to dispel any doubts.
- **Visibility** of the IT managers to the steering committee and also to politicians accentuates responsible action.
- The **monthly meetings** between all the unit project leaders are the main platform for exchanges among them and have now also become an important medium of communication with representatives of competing interests elsewhere.
- Support from the **migration coordinators** in the planning and preparation of the migration is an early guarantor of quality and commitment.
- The introduction of **release and test management**, partly at the recommendation of the unit project leaders, has substantially improved the reliability of our own planning and of the LiMux-Client.

- Support for **internal communication** from the LiMux core team helps managers whose skills are primarily technical to deal securely with 'soft topics'.
- In addition there is the experience that the **LiMux-Client** is working well on an everyday basis.

The IT managers and the technical leaders take on the role of central opinion formers and ensure that the migration is sustainable in their field.

3.6 IT staff: the 'initial hurdle' for the project

One of the first hurdles for the project was the involvement of IT staff in the migration fields. In some business units, it proved possible to persuade this stakeholder group of the merits of the project at an early stage. In other business units, on the other hand, there was some 'passive resistance', which could only be overcome in the course of the project. This resistance arose from loyalty to the familiar, locally optimised solution as against the central, city-wide LiMux configuration.

These IT staff have the role of technical managers within the project and act as the direct contacts of staff in the core team. Their tasks are as follows:

- to define the requirements with respect to the LiMux-Client,
- to act as test coordinators and receive the new versions of the LiMux-Client and
- to carry out the migration on the ground from the organisational and technical point of view.

The constant improvement of the LiMux-Client, the standardisation of server and administration tools, necessitated by the reorganisation to it@M, has played a decisive role in the growing acceptance of the open source strategy and solution. Increasingly, it is proving possible to turn this stakeholder group completely into fellow campaigners by means of:

- technical support for the migration on the ground by the core team,
- involving them in training and workshops and
- equipping them with the requisite know-how to act as administrators and user support officers.

3.7 Management: a difficult target group to gain access to

The LiMux project did not succeed in gaining direct access to the stakeholder group made of managers of the business units and their IT managers. Involving them was the task of the unit project leaders, but they were not equally successful everywhere. In many respects, this is highly regrettable, because on the one hand it denied the unit project leaders possible support in convincing other staff and on the other hand it meant that managers did not always act as positive role models for staff as one might have hoped.

Ultimately this was expressed in the form of:

- in some cases, excessive demands postulated by managers,
- failure to take delivery of the system,
- silent protests among staff and
- avoidable escalations above business unit head level.

The reason for this gap in the stakeholder approach is the lack of involvement of managers in roles and/or in regular communication. Efforts were made to overcome this problem by means of occasional visits to the regular management meetings.

From the point of view of stakeholder management, this stakeholder group adopts a position between 'expressing no opinion' and 'obstruction.' In addition, however, some managers opened up their minds for Open Source technology and used it not only when they were ordered to.

3.8 Ultimately it is the staff who make or break the project

The introduction of LiMux as a workplace system results in changes to the working environment for many staff. This confronts the City of Munich's staff with changes which entail uncertainties and grounds for concern. This concern is fed by two sources. On the one hand, the changes temporarily involve efforts, while on the other hand changes tend to generate uncertainties.

The temporary efforts arise from the fact that unfamiliar working procedures, or the same procedures carried out using unfamiliar programmes, simply take more time than those which have already become routine. Moreover, the know-how enabling staff to optimise their own work has to be built up afresh in the new environment. In view of the high workload to which staff are subject, one can understand their concern.

Changes, like anything else new, are viewed critically, which is hardly surprising: every business or large organisation experiences similar phenomena when it comes to replacing familiar things with something different whose advantages may first even have to prove their value before they can be accepted. Here, firm belief and perseverance are needed. Initially, many people reject the innovations, because change may make the professional future seem uncertain. But what ultimately matters is confidence and acceptance on the part of staff. Without their support, successful implementation would be inconceivable.

An important message to IT staff had to be that IT development and the migration to the new office suite and to the LiMux system are not an end in themselves but yield demonstrable benefits. Staff must be involved. It is therefore vital to generate acceptance of the changes. That will not happen automatically: it requires active support in an ongoing process.

In order to take this into account, the migration is being accompanied by management of change (to give it its full title, 'management of change and communication': see also Section 3.1, 'The project team: the starting point for all stakeholder groups'). This involves two team members from the project management level who are specifically working on internal and external communication (see also Section 3.10, 'PR work: a two-edged sword') and support in the process of change.

On an everyday basis, this can mean that, long before the migration, info tours are already being made through the relevant fields, which entail users and IT staff alike receiving information about the impending changes. Shortly before the migration, moreover, further meetings are held in some fields, which receive support from the 'management of change and communication team' or also, occasionally, from the project management on the spot, jointly with the IT staff. In addition, when further meetings are held, those who have this responsibility repeatedly drop in or participate in working parties/regular discussion days where, as a rule, the IT staff from all migration fields are represented.

The LiMux team can be contacted quickly by e-mail, using an internal communication address, and will also on occasion be present at one gathering or another. Mostly, however, contact is maintained by means of exchanges with the relevant LiMux communication officers in the fields.

The team „***M****igrations****u****nterstützung* ***v****or Ort*“ (Migration Support on the Ground), with its acronym MUV, supported the migration fields from the technical point of view on the ground: in order to prepare for the migration, MUV invested two days of preliminary work in the offices of the business unit concerned. If there are questions or problems after the migration, the team again goes to the scene and helps to find solutions. In addition, there are regular review deadlines.

Enormous value is attached to the training of staff: during the migration, first of all the IT service staff and IT user advisers are trained, after which they are constantly updated about the latest situation. They then become multipliers and ambassadors for the end user, as they are the initial contact and trusted individual. If the migration has the support of those in charge of IT services with the support of the business units, their staff will mostly go along with it. Each employee of the city of Munich was also entitled to half a day's training with the new operating system and up to a day for the Office package. For those who wish to gain more in-depth knowledge, there is the 'LiMux-Lernwelt' for individual study.

The LiMux brand is always to the fore. With the penguin Tux, the mascot of the free operating system Linux, LiMux has its recognisable identity at every meeting. The penguin soft toy, customised with the Munich 'Kindl' coat of arms, helps people to identify with the product and with the LiMux brand, and other small merchandise such as LiMux desk pads is well received and creates a little bridge to the end-user, thus assisting identification with the new system.

Of course it is advantageous if one can manage to turn the bulk of the staff into fellow campaigners too. But it is already sufficient if one ultimately succeeds in persuading staff not to adopt a hostile attitude towards the new work station but simply to accept it as an aid in the ongoing provision of services to the public. The most important point for the staff is that the LiMux client provides the appropriate business functionality and stability. This is guaranteed by the LiMux Client. Therefore, the client is accepted.

3.9 The open source community: a reliable fellow campaigner and developer

Whether it is through the voluntary cooperation in the further development of open office communication suites or the organisation of developers' days, the commitment displayed in associations which support the open file format is a wholeheartedly espoused principle of the open source approach.

Munich is a pioneer in the large-scale use of open-source software, but it is also active in the further development of that software in the context of associations such as OpenOffice.org or its independent continuation: LibreOffice. It gains much in return for this:

The open-source community is large and closely linked. From time to time, meetings are held, which are not at all virtual but physical. For example a 'hackers' party' or a 'bug-squashing party', where developers from all over the world meet and devise solutions to specific problems with programmers from the LiMux project. The result is then amalgamated with the software solutions.

There is also increased cooperation between IT staff representing various public institutions to deal with compatibility issues across the board. Examples include the OOXML Workshop of the Swiss Open Systems User Group (http://www.ch-open.ch), in which, in addition to Freiburg, Jena, Munich and the Swiss Federal Court (among others), various community organisations and a representative of Microsoft also took part. Work packages were adopted to improve support for the standard developed by Microsoft for its OOXML file formats. The workshop represented an important step forward in cooperation between municipalities. The features to improve interoperability are implemented and available starting with Libre Office release 3.6 and were further improved with Libre Office release 4.0.

The LiMux project is an active member, or at least contributor to the following groups, among many others: the OSBA (Open Source Business Alliance e.V.), FrODeV (Freies Office Deutschland), OpenOffice.org, LibreOffice, TDF (The Document Foundation), FSFE (Free Software Foundation Europe).

Fruitful exchanges are organised by means of mailing lists, for example concerning the template system WollMux developed by LiMux itself, and concerning Ubuntu (a free and open source operating system) and other open-source projects. WollMux was developed as an extension of OpenOffice.org and is now being used in more and more municipalities for work with templates, forms and letterheads (www.wollmux.org).

The open source community is in a twofold sense a readily accessible source of strength for a project such as LiMux: without the uncomplicated and rapid support of the community, a project of this kind would not be feasible. The selfless, uncommercial attitude of the open source community to sharing knowledge and know-how is the guarantor of the success of the whole open source movement.

3.10 PR work: a two-edged sword

Successful internal and external communication (see also Section 3.8, 'Ultimately it is the staff who make or break the project') also requires PR work. This includes contacts with the press, involving interviews, podcasts and articles, as well as participating and speaking at trade fairs and congresses. Talking about the subject and reporting on it are part and parcel of this. Staff have a right to expect it, and an interest in seeing how the project is viewed and assessed by the public. They, after all, are also part of the project and should be informed about the impression that the 'flagship project' is making on other interest groups. Thanks to transparency regarding the progress of the project, other municipalities may perhaps feel inspired to consider migration. That is also the intention of the project: to organise exchanges and gain fellow campaigners. If the EU institutions were to be convinced to opt more for open source technology in future, it would help if the public were behind it too.

PR work naturally also gives rise to expectations and leads to new challenges: press work is difficult to control, as an editor – an independent authority – may intervene. It may sometimes be difficult to influence the nature of the publication, which means that it soon becomes necessary to issue corrections, which take up time and energy that would otherwise be available for other developments. A posting on the city's IT blog or on the professional blogs of various online IT platforms will elicit all manner of comments. It is therefore vital to be vigilant in monitoring the news.

While public exposure opens one to attack, it is clear what advantages can accrue to the project from providing information. LiMux therefore has an up-to-date presence not only on its own staff website but also on the public Munich Portal (www.muenchen.de/limux and www.it-muenchen-blog.de). The annual goals and milestones of the project are also indicated there. Contributions on Wikipedia are kept up to date and contacts with other municipalities are gradually being established and strengthened.

A measurable result of this public profile can now be seen in the form of the substantial numbers of requests which have been received from associations or towns seeking support in deciding to migrate to free software. Again and again, the LiMux project team is sounding out possibilities of further propagating the open source idea. For example, it plans – jointly with those who run the city – to lobby more fervently for the adoption of an EU directive on the exclusive use of open standards. In future, other urban partners will also be invited to cooperate more. In other words, lobbying also forms part of PR work. In addition, the city of Munich hosted its first public Open Source event in the city hall at the 20th and 21st of June 2013. Agenda and presentations are available for download on www.muenchen.de/opensourcetage. The event was held in German.

The social media are not yet playing a major role, because here too, it is necessary to act effectively and to make the right impression: otherwise, far from producing added value, these media will lay the authority open to further attacks. And, as everywhere else, in Munich too resources are at a premium.

Openness calls for transparency, and transparency is always a basis for acceptance. As acceptance is ultimately what matters, there is no alternative to open communication. The general public constitutes a diffuse stakeholder group, which cannot be assigned a blanket position in the force field analysis. Rather, it is necessary to consider the individual subgroups separately and to communicate with them in a targeted manner.

4. LEARNING THE LESSONS OF THE PROJECT TO DATE; PROSPECTS FOR FURTHER DEVELOPMENT

Such a far-reaching IT transformation process, requiring acceptance, as the one that took place in Munich, is only possible if vital stakeholder groups cooperate (Moser 2007). Stakeholder management has ensured the success of the LiMux project in Munich. The relationship with each stakeholder group needs to be built by means of different measures.

NB: The points below are intended not as recommendations but as indicating the outcome of the learning process during the project:

- The core team is the prime mover of the project and the origin of its success. Staff belonging to the core team regard themselves as promoters of the open source community.
- The City Council is a fellow campaigner and sponsor of the project and, as a messenger with a mission to the public, requires up-to-date, open communication about the progress of the project.
- The Umbrella Staff Committee and its business unit Staff Committees are an influential and competent stakeholder group, whose involvement is particularly necessary at the beginning of the project.
- The IT management team acts as a promoter of the project, always keeping an eye on the business units' practical constraints. Together with the City Council, it constitutes the project governance structure.
- The IT managers and technical leaders ensure the sustainability of the migration process and must therefore be encouraged to become fellow campaigners or even promoters of the project.
- IT staff are the Achilles heel of the project: unless there are plenty of staff with open-source know-how, a lasting changeover to open source technology is not possible.
- Management must be thoroughly involved in the role-based structural model, in order to set an example to staff: otherwise, there is a danger of impasses and passive resistance.
- Staff are satisfied that the new work station is functioning as an aid to the ongoing provision of services to the public.
- The open source community is a highly accessible source of strength for such a project. Exchanges of know-how and knowledge on a not-for-profit basis are a guarantor of the success of the whole open source movement.
- Openness calls for transparency, and transparency is always a basis for acceptance. As acceptance is ultimately what matters, there is no alternative to open communication, including with external stakeholder groups.

Upon closer inspection it becomes clear that, in conjunction, the City Council, the IT management, unit project leaders and the project team are the driving and stabilising forces in the whole transformation process. In the course of the project, three positions, in particular, have changed fundamentally: the Staff Committee, after playing a more active role in the early stages of the project, is now less important, while the unit project leaders and IT staff have become significantly more positive about the Limux project. The attitudes of the other stakeholders have largely remained unchanged. With regard to quality, too, the stakeholders are not all alike. For example, management and IT staff are very heterogeneous: there are almost equal numbers of strong advocates and critics. The project team and also the IT management, on the other hand, each constitute a very homogeneous group. Contrary to some rumours, the instigator of the project and political authority – the City Council – has not changed its attitude: it has been a strong promoter throughout. Over the years, it has proved possible, over all, to persuade some stakeholders to abandon the position of wishing to express no opinion and to involve them in the project.

In 2013 this project is going to be finished as a project and the responsibility for the development, release management and maintenance of the LiMux client is going to be part of the regular business line. The IT of one of the largest municipalities in Germany is now independent, free and modern. Munich's IT development regarding the LiMux client is completed as a project. The LiMux project has proved equal to its role as a 'flagship' project. Now it only remains for many other municipalities, authorities and organisations to follow its lead.

5. BIBLIOGRAPHY

- Acohido, B. USA TODAY, 13.07.2003: http://www.usatoday.com/money/industries/technology/2003-07-13-microsoft-linux-munich_x.htm
- Freeman, R.E. (1984) Strategic Management. A Stakeholder Approach. Pitman, 1984
- Lang, C. (2010) Die Stakeholderanalyse im Rahmen des projectmanagements.
- Moser, P. (2007) Stakeholdermanagement zur optimalen Gestaltung strategischen Wandels
- Theuvsen, L., Schauer, R., Gmür, M. (2010) Stakeholder-Management in Nonprofit-Organisationen.

EUROPEAN PARLIAMENT

CATALOGUE : BA-01-13-231-EN-C

DIRECTORATE-GENERAL FOR INTERNAL POLICIES

POLICY DEPARTMENT C
CITIZENS' RIGHTS AND CONSTITUTIONAL AFFAIRS

Role

Policy departments are research units that provide specialised advice to committees, inter-parliamentary delegations and other parliamentary bodies.

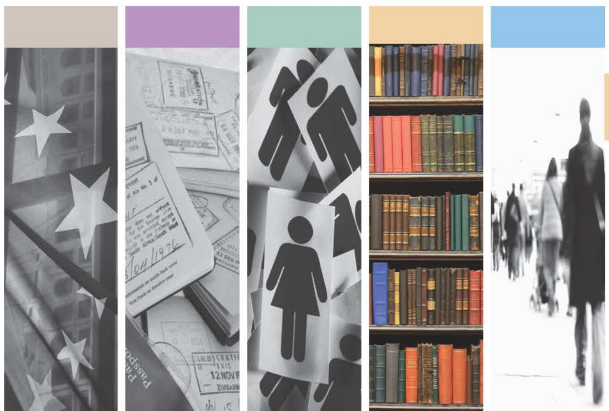
Policy Areas

- Constitutional Affairs
- Justice, Freedom and Security
- Gender Equality
- Legal and Parliamentary Affairs
- Petitions

Documents

Visit the European Parliament website: **http://www.europarl.europa.eu/studies**

PHOTO CREDIT: iStock International Inc.



ISBN : 978-92-823-4570-2
DOI : 10.2861/21189

Publications Office