Free-Libre Open Source Software as a public policy choice

Mark Perry*
Faculty of Law - Faculty of Science
University of Western Ontario
London, Ontario
mperry@uwo.ca

Thomas Margoni*
Faculty of Law - Faculty of Science
University of Western Ontario
London, Ontario
tmargoni@uwo.ca

***Abstract*—Free Libre Open Source Software (FLOSS) is characterised by a specific programming and development paradigm. The availability and freedom of use of source code are at the core of this paradigm, and are the prerequisites for FLOSS features. Unfortunately, the fundamental role of code is often ignored among those who decide the software purchases for Canadian public agencies. Source code availability and the connected freedoms are often seen as unrelated and accidental aspects, and the only real advantage acknowledged, which is the absence of royalty fees, becomes paramount. In this paper we discuss some relevant legal issues and explain why public administrations should choose FLOSS for their technological infrastructure. We also present the results of a survey regarding the penetration and awareness of FLOSS usage into the Government of Canada. The data demonstrates that the Government of Canada shows no enforced policy regarding the implementation of a specific technological framework (which has legal, economic, business, and ethical repercussions) in their departments and agencies.**

***Keywords*-Public policy, data analysis, derivative works, licences, public sector, policy recommendations.**

I. Introduction

Free-Libre and Open Source Software (FLOSS) is about freedom, access, transparency, and accountability [1]. However, at many levels FLOSS is considered for use by provisioning departments only on the basis of the asserted cost reductions that it may bring. This is reflected in the fact that the acronym is sometimes replaced by other non-standard terminology. For example, the Canadian Government recently made a 'No Charge Licensed Software' Request for Information, thereby avoiding the FLOSS terminology [2]. Although terminology may seem like a minor issue, many advantages of FLOSS are found in software that fits strictly within its definition, and not within acronyms and labels that could look similar to a non-discerning reader. Examples of this include 'free-ware', 'share-ware' and 'no charge software'.

In this paper, we look at the attributes of FLOSS that are linked with the specific legal requirements under which it is distributed, as well as the specific framework under which the software is developed. We focus on the public sector, i.e., government, public administrations, federal or provincial agencies. FLOSS is also successfully employed in the private sector, however, we have not analysed the idiosyncrasies of the corporate and business initiatives [3]. The following sections are included: basic terminology; licence compatibility; benefits for governments and public agencies to adopt FLOSS, and whether such beneficial aspects are caught by the Government of Canada (GoC). Finally, we present the data of a recent survey that suggest that the GoC is not taking advantage of the benefits this innovative phenomenon would include.

II. Terminology

When dealing with FLOSS, two main concepts of the taxonomy should be separated: "Free-Libre" and "Open". There have been disputes between these two approaches, which lie behind similar code development methods: "[T]he obvious meaning for the expression 'open source software' is "You can look at the source code", and most people seem to think that's what it means. That is a much weaker criterion than free software, and much weaker than the official definition of open source. It includes many programs that are neither free nor open source. Since the obvious meaning for 'open source' is not the meaning that its advocates intend, the result is that most people misunderstand the term" [4].

Opposing the Free Software Foundation (FSF) position is the Open Source Initiative (OSI) position: "it was decided at a conference that it was time to drop the moralising and confrontational attitude that had been associated with free software in the past, and sell the idea strictly on the same pragmatic, business-case grounds that had motivated Netscape. They brainstormed about tactics and a new label Open source, contributed by Chris Peterson was the best they came up with" [5].

Despite advocates of different points of view towards these positions, and the different weight that each position gives to ethical and business concerns, the difference lies more in a philosophical level of abstraction rather than at a substantial level. Further, if it is true that Open Source is appealing for the private sector, then in the public sector (whose primary goal is not to make profit) a definition that includes 'Freedom' seems to fit perfectly.

However, more important than definitions, when a provisioning department is deciding the type of software that should be used it must look at the legal requirements that establish the terms of use of the software, i.e., the licences.

A. Licences

The first and most famous of these agreements is the GNU General Public License, (GNU GPL or simply GPL). The first version of the license was established in February 1989 and was quickly followed by a second version in June 1991. The wording "version 1 (or 2, or 3) or any later version" is often used. The current version 3 dates back to June 2007. Between 50% and 70% of all FLOSS projects are released compliant with one of the versions of the GPL [6]. Although the GPL covers more than a half of all major FLOSS projects, there is a plethora of other licences that are commonly used. The Open Source Initiative (OSI) reports 55 OSI approved licences. The Free Software Foundation (FSF) reports 43 Free Software approved licences compatible with the GPL and 39 licences that are deemed Free Software compliant but not GPL compatible.

In many cases, to be GPL compatible is a matter of version and different versions of the same licence can be either GPL compatible or not. It is interesting to note how, technically speaking, version 2 (v2) and version 3 (v3) of the GPL are not compatible with each other. The issue of compatibility is of paramount importance when a public agency decides to develop its own software tools, either because it must be aware of the licensing already set up for existing code, or if they want to create new code, or open-sourcing an existing one, they need to be aware of the differences, opportunities and limits connected with different licences.

B. Updating the GPL

While GPLv3 aims to maintain and further the principles of GPLv2, technological pace and new threats to FLOSS obliged GPLv3 drafters to insert specific provisions to address these new issues. The drafting of v3 was a socially distributed effort and many criticisms that emerged on early drafts regarding new requirements have found their place in the current version. In particular, besides a general improvement in terminology, especially for internationalisation [8], and a more detailed section on definitions, v3 contains the following new sections: Digital rights (or restrictions) management/TiVoization (sec. 3); Licensing patents (sec. 11); Short term compliance (30 or 60 days) before automatic termination of the licence (sec. 8); Clarification regarding peer-to-peer distribution of object and source code (sec. 9); And a less monolithic general framework (sec. 7).

Unfortunately, since both v2 and v3 are strong copyleft licences (see *infra*), this excludes their mutual compatibility. If a work is released under GPLv2 and somebody wants to distribute a modified version of this work, this modified version shall be under GPLv2. The same happens with GPLv3. The problem arises when a modified version is based on more than one program, where at least one is under v2 and the other under v3; in a case like this, there is no legal way to make them compliant.

However, it must be kept in mind that the copyleft requirement applies only to modified versions. That is to say, it is possible to distribute a software package (e.g., an operating system) containing programs under different licences, even those not compatible with each other. This is what happens with the most common GNU-Linux distributions, where the Linux kernel distributed under GPLv2 happily coexists with other tools or applications released under GPLv3 (or many other non-compatible licences). Cases like proprietary Loadable Kernel Modules (LKMs) or binary blobs, i.e., those object files loaded into the kernel without a publicly available source code, are considered borderline cases, meaning that in some limited circumstances their use is accepted because it is recognised that they do not form a derivative work of the kernel.

GPLv3 may be combined with important licences that are not compatible with the former version, namely the XFree86 (v. 1.1), the Apache (v. 2.0) and the GNU Affero GPL (v.3) licences. In particular, the latter is a GPLv3 licence specially aimed for network-interactive software, thus allowing users of web-applications to be able to receive the source code (technically speaking, to run a server is not an act of distribution).

C. BSD: licence and versions

A criticism of FLOSS licence regimes is as to the naming system. Law requires certainty in many aspects, including terminology. If versioning in regards to the GPL licence sounds confusing, then the Berkeley Software Distribution (BSD) licence offers a much more challenging example.

BSD is a FLOSS licence (FSF recognises it as Free Software) but it is a permissive licence, meaning neither strong nor weak copyleft (see *infra*). The BSD licence should, more correctly, be referred to as an entire family of licences, rather than only one. The main reason for this classification is the multiple modifications that the original licence has suffered, thus when software is distributed with a BSD licence, it is of pivotal importance to know the exact version.

The *new*, or *revised*, or again *3-clause* BSD licence is clearly Free Software and GPL-compatible. However, this compatibility does not exist when referring to the *original*, or *old*, or *4-clause* BSD licence. In the latter, an extra clause imposes a requirement that makes it incompatible with GPL. This clause, also called the 'advertising clause', requires authors of derivative works to include an acknowledgement of the original source, which, could lead, and sometimes has, to many pages of acknowledgements. Each of these sets is basically composed of the same licence with slight variations in the wording.

In addition to these two main categories, the BSD family has grown. Among the more widespread, there is the NetBSD, the 2-clause BSD (similar to the MIT), the FreeBSD, and the Clear BSD. All of these variations of BSD are usually GPL compatible, though this does not mean that their actual wording should be ignored. On the contrary, it is important to know, for instance, if the licensor reserves the right to sue you for patent infringement or not (see the Clear BSD).

III. Copyleft's reach

In certain production circumstances the use of some types of FLOSS licences are perceived to be problematic. Some creators and distributors of 'packaged' software have detracted from GPL due to its so-called viral nature. The word 'viral' is unfortunate, as it projects a negative connotation upon a clause in a legal document. To see such a characteristic with favour or not is a matter of personal choice, but as a matter of legal definitions, it should be referred to with a more neutral epithet: here we will refer to this characteristic as 'persistence'.

A. Strong copyleft

One of the characteristics of the GPL is its strong copyleft status. Strong copyleft licences are those licences that require any subsequent distribution of the work, or a modified version of the work, to be under the same licence. A new program based on a GPL licensed code must be distributed under the GPL. This persistence has represented a major issue in the field of FLOSS. Some supporters of FLOSS models that are based on non-persistent, but permissive licences (i.e. similar to the BSD), have accused the GPL of cannibalising BSD software: while the permissiveness of BSD-like licences permits protected code to fall under the GPL, however, the converse is not possible due to the copyleft requirement of the GPL. GPL supporters argue that it is not the GPL cannibalising code, but rather the BSD that permits every type of licence and even propertisation of BSD software. Copyleft, proponents say, is necessary to protect and foster the development of a "contributory commons".

B. Weak copyleft

There are some FLOSS licences that are copyleft but their requirements are not as strong as the GPL. Consequently, they are labelled 'weak copyleft' licences. Examples of this category are the LGPL (where 'L' stands for 'Lesser'), the Common Public Licence, and the Mozilla Public Licence. These licences allow combining the software with other types of licensed software without the necessity of distribution under the same licence, but this does not mean that they don't need to be compatible: the CPL and the MPL, unlike the LGPL, are not GPL-compatible [17][18]!

The difference between weak copyleft and permissive regimes is the possibility to combine, for example, LGPL and closed-source software without turning the output into LGPL. However, such a feature applies only to linking activities. If a piece of software released under the LGPL is going to be modified in order to produce a new version or a fork (or every other activity but linking) the new software will have to be released under the same licence (or eventually the standard GPL), thereby fulfilling the copyleft part of the label. Since persistence only works for some types of activities and not others (linking in LGPL case), such a copyleft regime is not strong, but weak.

C. Derivative works

This brief overview of the compatibility issues regarding weak copyleft licences necessarily brings us to the concept of the derivative work. The aim of this paper is not to provide an exhaustive analysis of what this concept could legally mean, as, due to the ubiquitous nature of the Internet, such a survey would have to be completed for all jurisdictions. What we analyse is the meaning of derivative works in the case of a program being linked by another one, usually a library, and observing the unique consequences derived from the wording of the GPL in cases of dynamic linking and static linking, and ultimately whether this distinction does, or should, matter.

A program statically linked with a library, creates a new, modified work. If either piece of software is released under the GPL, the derived work (the program statically linked with the library) shall be under the GPL. Since part (a 'substantial part') of the library is copied into the executable of the program at compile time, the output is the program plus the library (substantial part thereof), and thus a new work based on the two precedents. If one of the two works is released under the GPL, the new derivative work will have to be under the same licence as per the GPL requirement.

A more complicated case is that of a program that is dynamically linked with a library. In such a case, no substantial part of the library is present into the executable, so besides being connected, the latter is not a derivative work. However, while the FSF and GNU agree with this general framework, they further affirm that when a dynamically linked library and program share a more 'intimate' existence, they should be considered once again a derivative work. More precisely "[i]f the program dynamically links plug-ins, and they make function calls to each other and share data structures, we believe they form a single program, which must be treated as an extension of both the main program and the plug-ins, while if the program uses fork and exec to invoke plug-ins, then the plug-ins are separate programs, so the license for the main program makes no requirements for them" [12].

A complex, borderline case, where in presence of a dynamic linking structure the FSF and GNU support the thesis of a derivative work (extension of the two codes), due to the relation between the two pieces of software, which is so strict (reciprocal function calls, sharing of data structures) that, even in the absence of a substantial portion of the source code of one of the programs into the other, the functional result is not far from it. Nonetheless, FSF and GNU recognise the presence of undefined areas: "If the program dynamically links plug-ins, but the communication between them is limited to invoking the main function of the plug-in with some options and waiting for it to return, that is a borderline case" [12].

The Canadian Copyright Act, for example, gives little guidance for such situations (as is common in almost all jurisdictions), only generally reserving the right to "produce or reproduce the work or any substantial part thereof in any material form whatever... and to authorise any such acts" to the rights holder, and adds specific cases of adaptation, that are inapplicable to software (sec. 3, especially d. and e.). The Act is in much company with the United States and many European countries, as the legislation does not deliver a granularity that is fine enough to deal with a library dynamically linked to a program with which it shares system calls and data structures. This is probably the better situation as the legislation is meant to provide general and abstract rules, leaving it for the interpreter to adapt them to a specific case. Here, it might be relevant to recall that in Canada, while rewriting a computer program from one language into another could be interpreted as a translation under certain circumstances [13], compiling the source code into object code is an act of reproduction [14]. The main consequence of this distinction is that to compile a program (being either an application or library) requires the right to reproduce [15].

Determining exactly what a derivative work is within linked computer programs is a contentious issue. It obviously depends on the legal system where one claims protection. However, there are claims that the issue of static and dynamic linking is a red herring and what really matters is not the name of a specific program or call (mkisofs, ld, exec, or the like), which undoubtedly has functional consequences, but the specific grade of dependence or independence between the two programs. This relationship establishes whether the output is a derivative work or a mere aggregation [16]. The latter approach introduces some uncertainty because it suggests a case-by-case analysis, rather than a "static = derivative" equation. The door is still open to deeper analysis on this issue, as is evidenced by the comments of one of the fathers of the Linux kernal, Linus Torvalds, when he said that 'there was not much need for the LGPL' [16].

IV. Not just Money

Usually, obtaining FLOSS requires nothing more than an internet connection. Inherent in both the FSF and OSI models is the ability for anyone to access the code. There are no royalties to be paid, no required tie-in to service contracts, and no up-front acquisition costs. In addition to the economic aspects, there are many advantages to adopting FLOSS: although price is not the primary advantage, it is often viewed as such, which results in FLOSS being incorrectly assimilated with other non-immediate-fee software. Such naivety should be avoided, especially when the interested entity is a public body whose main objective is to offer public services and not to make a profit. An important aspect of FLOSS is the availability of the source code. This means that the ability to modify and redistribute improvements is a contractual obligation. This specific feature is common to all licences fitting in the category and therefore entails legal, economic, technical and social consequences. We will explore nine examples of these consequences, which are particularly pertinent to governments' use of code.

A. Accountability and transparency

Source code availability permits users to know what the program does at a depth that would otherwise be impossible. Without the source code one can only deduce what the program does through expensive and time consuming reverse-engineering without ever having the opportunity to know all of the original code. Source code availability is critically important for software applications in the core areas of government (such as national defence and homeland security, financial and economic administration, health databases, and wherever privacy and reliability are deemed substantial), as well as the fundamental infrastructure of public administration [7]. The possibility for the general public to understand and to rely on the activities of public bodies is directly connected to the use of a software model that is transparent and accountable (e-Democracy). This is a cornerstone in providing citizens with the guarantees of a fair, efficient and impartial administration of the public good. A good example of this can be seen in electronic voting systems [9].

B. Interoperability

The availability of the source code allows for better interoperability with other applications. If an application is not perfectly compatible, the availability of the source code, combined with the contractual permission to use and adapt it, permits modifying the code with interoperability as the likely result. If there is FLOSS and closed source software (proprietary), greater compatibility is possible in contrast to the case of two closed source software provided by different suppliers [11]. This is of particular interest for public bodies since it grants the possibility to share resources among many different departments and agencies. Despite being autonomously organised, public bodies do not suffer from the strict competition that affects corporate entities. This is what allows for strong scale economies with significant savings for the whole public administration and, consequently, for taxpayers. In some jurisdictions (e.g., Italy) this is prescribed by law [10].

C. Avoid lock-in

Vendor lock-in is the phenomenon that causes customer dependency on a given vendor with regard to a specific good or service. Switching vendors has high transaction costs connected with technological and organisational changes and, in some cases, penalty clauses due to early cancellation of a supply contract. These 'switching' costs are pernicious to the market and can represent strong barriers to entry. With closed source software the customer is generally bound to a specific supplier, both contractually and technologically. As an example, in the case of freeware, a typical business model that is sought is lock-in. In this case, once the lock-in has occurred the software distribution model can switch to a traditionally priced one since the transaction costs connected with the migration to another type of software are prohibitive [35]. In the case of FLOSS, both the licence and the technology allow for a supplier-independent business model [19]. For public administrations, it is mandatory to choose suppliers that are able to grant reliable services at good prices and provide for long-term maintainability (public administrations usually last longer than private companies). However, it is also critically important that if a better offer or player enters the market the public body should not be impeded from transitioning to the more efficient solution. This will immediately reflect in the cost and quality of service enjoyed by citizens.

D. Long-term maintainability and technological ecumenism

A public administration cannot discriminate the public based on the type of software used. A private company has the option to use closed source software compatible with 85% of the software used by citizens and incompatible with the remaining 15%: the market will decide if this decision pays. However, a public administration cannot exclude 15% because they chose a different operating system. FLOSS is the solution that grants the highest compatibility, thus minimising the phenomenon of technological exclusion by both FLOSS users and closed source software users. FLOSS also means Open Formats, which are those formats that are publicly documented so as to permit anyone to implement programs (both FLOSS and closed source software) that can optimally use, store, and retrieve such data. This is another manifestation of the absence of lock-in problems [20].

Many times the reason for staying with an old supplier (which usually means also old technology) is that they are the only ones owning the (closed) format technology enabling data retrieval.

E. Security and error correction

Security is not a static concept that can be reached once for all, nor easily maintained. FLOSS is known not only for the transparency and accountability of its code, but also for its stability and intrinsically greater security. It is a common principle in computer science that the security of a system depends on the quality of its structure, not on its obscurity (a variation of Kerckhoffs' principle in cryptography). Only if the source code is available is there the possibility for quick bug-correction and exploit-detection. In the case of FLOSS, the pace at which the stability level of the code grows is much faster than in other types of software, where it is necessary to wait for the supplier security updates [21]. A sound and accountable technological infrastructure is a key point for all e-Government and Government-to-citizens (G2C) initiatives, where the reliance of citizens is fundamental for the success of the electronic offered service.

F. Democracy and pluralism

FLOSS in the public sector is more generally a matter of democracy [22]. In case studies such as those involving electronic voting machines, or "technology enhanced trials", the people need to rely and trust not only in their representatives and the courts, but also the process of electing the candidates or of condemning the guilty [22]. FLOSS seems to epitomise those basic principles commonly found in many constitutional and fundamental charters, of fair administration of the justice, of pluralism, of freedom of expression, and of access to information and knowledge. A long list of public administrations around the world has already started, or is seriously considering migrating from proprietary to open code. Among the most successful initiatives is the German city of Munich with the LiMux project. They report to already have 1,200 workstations migrated to Gnu-Linux, 12,000 using Open Office, and 100% of the city administrations using Firefox and Thunderbird [33]. Another interesting case study can be seen in Spain, where the different comunidades autonomas (Spanish provinces) have different levels of FLOSS implementation, all coordinated by a specific constituted public agency: Centro Nacional de Referncia de Aplicacion de las TIC basadas en fuentes abiertas (www.cenatic.es). The "dollar price" connected with the absence of royalties is only one potential saving: "Contrary to what is often assumed, cutting costs was not the main reason for the migration. The motivation is independence [...] now we're able to decide on our own how we want to spend our IT budget in the long run [...]" [23]. This approach is also consistent with the Open-Government instances which hold that the business of government and state administration should be opened at all levels to effective public scrutiny and oversight. To translate the Open Government principles in programming terms, involves the use of FLOSS.

G. Portability to other languages

The possibility, both technical and legal/contractual, to translate software into any language is of paramount interest if due importance is given to linguistic and cultural pluralism. Although this sounds more like a European, Asian or African based argument, also in America (both North and South) language plays a key role in the protection of indigenous and traditional knowledge and in effectively reducing the phenomenon of 'digital divide'. This feature of FLOSS may be easily confirmed by checking the language packs or language ports of some of the most widespread projects and comparing them with similar non-FLOSS products. For example, Firefox v3.1 has 62 language ports and 78 different language packs, among which many minority languages are present. On the other hand, Internet Explorer 8 has 3 language selection possibilities. Opera has 41 supported languages, Chrome 44, and Safari 18. The difference is even greater with the office suite: OpenOffice.org has 123 supported languages, while Microsoft Office has 35. Finally Outlook 2007 is available in 14 languages, while Thunderbird 2, in 39.

The reason for this difference in language policy clearly resides not only in the sensibility of the project managers but rather in the declared legal, contractual and technological features of FLOSS.

H. Fostering competition

Another major advantage of FLOSS is that it creates and favours a more competitive ICT environment usually populated by many local Small and Medium Enterprises. Licence fees, from a microeconomic point of view, represent huge barriers to entry the markets, thus favouring monopolistic and oligopolistic situations. As it has been reported [23][24], a public administration investing in FLOSS solutions is usually interested in hiring or contracting with local ICT companies for services like updating, maintenance, training, and customisation. In this way the immediate benefit for local economies is apparent.

I. Total Cost of Ownership

A major saving in using FLOSS is royalties. Quite simply, there are none. During the 2005-2006 fiscal year the Canadian government spent 425,602,327CAD on software licence fees [25]. Clearly, this represents a huge amount of money. Unfortunately, using FLOSS does not mean that there are no costs whatsoever. For example, due to the so-called *alumni effect*, many people have learned how to use computers through non-FLOSS applications. This means that even though FLOSS solutions nowadays are user-friendly enough, there are still some costs connected with migration, such that in the short term, it is not always true that there are significant monetary savings. Nonetheless, there are savings that become substantial in the medium/long term and that will endure and increase with time. Some of these savings have already been identified (no lock-in, enhanced security, etc), while others are more concealed (such as cross-platform availability, maintenance, updating and long-term upgrading, compatibility with 'older' hardware, etc) [26].

Taking into account all of these variables provides a better portrait of the actual benefit in terms of economic and financial costs. As demonstrated in many studies, the huge Total Cost of Ownership savings resulting from the use of FLOSS is undisputed. The public sector reports from Sweden show yearly savings of billions of dollars [27]. Another benefit is that the agency taking the FLOSS route will need to spend money on the development of internal staff skills, which means that the skill base for the organisation will be improved, giving better overall support for the department and creating a greater pool of skilled persons in Canada.

V. The Survey

During this research, we conducted a survey on the use of FLOSS by the different Canadian ministries and other Canadian public departments and agencies. We contacted a total of 53 Information Technology (IT) departments. We decided to only target the category of IT departments in the agencies, since this allowed us to access the real technological situation of the department. In this survey we are not interested in what people do, whether given Canadian civil servants use or not FLOSS. Our survey was focused, and our data demonstrate, the use of FLOSS in Canadian governmental and other public agencies departments. Of the 53 IT departments contacted, 20 agreed to participate in our survey, either by live interview, phone interview, or through email. We preferred the live interview because it allowed keeping track of more variables than what appeared on the answer sheet. Interviews also allowed the operator to record the immediate reactions to the questions, which was not possible when using an emailed questionnaire. The participation rate was 38%, which although not very high, places itself at the top average of similar studies. For example, samples of reference participation data are 23.8% in the UK and 18% in Germany, without subdividing by the sector. A slightly higher participation rate is observed if considering only the Public Sector (37% and 29%, respectively). It must be noted that a third surveyed country, Sweden, has much higher participation ranges in every sector at approximately 60% [28]. Another seminal study in this field, Flosspols, reports an average participation of 22.8%, even though the variations from country to country are very high [29].

We had hoped that the participation rate to be higher than what was achieved because our respondents were public administrations, public departments, and ministries of the Canadian government and, as such, we stressed our identity (a renown Canadian University) and the fact that the study was funded and promoted by an important Canadian public agency (Social Sciences and Humanities Research Council, SSHRC). Unfortunately, this proved to be an incorrect assumption, as our viewpoint was not widely shared. The questionnaire was formed by 11 multiple-choice questions. The 12th question was left open so that the respondents could add whatever they deemed important that was not covered in our interview.

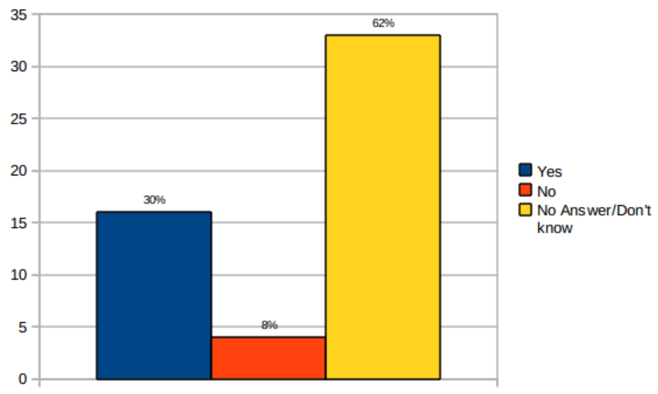

Figure 1: Do you use any Free-Libre Open Source Software in your department?

The results show that FLOSS in the technological infrastructure of the Canadian government is used only partially. In fact, its use is limited and not exclusive to what is referred to as 'Desktop purpose machines', but rather is utilised only in a very limited amount of cases on servers. Regarding Desktops, the fact that it is used only partially may be easily explained by a simple consideration. While it is uncommon to have non-FLOSS applications running on FLOSS Operating Systems (OS), the contrary is quite common. Such an inference is confirmed by the results of our next question, regarding the type of FLOSS (identified by name) used by the respondents.

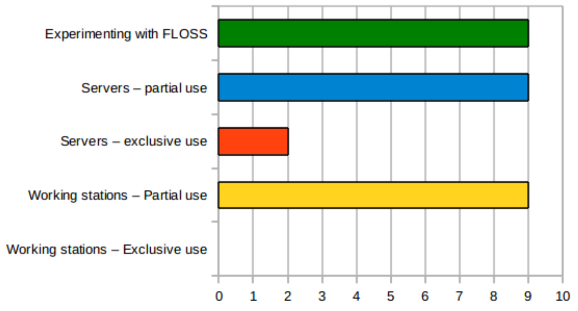

Figure 2: What type of FLOSS is it in use in your department?

Among the programs under consideration, Mozilla appeared most frequently (our questionnaire includes all the different varieties of projects that such a brand encompasses: Firefox, Thunderbird, SeaMonkey, Camino, Fennec, etc.) with 17% of the respondents reporting its use. Others widely used FLOSS programs included administration and database management or programming tools such as PHP (16%), Pearl (13%) and MySQL (14%) and interoperability tools such as Samba (8%). OpenOffice.org was used by only 5% of respondents, which might be explained by compatibility issues and *alumni effect* (see *above*). Among the less used programs were graphical desktop environments such as Gnome (6%) and KDE (5%). Graphical desktop environments are those programs used to provide users with a Graphical User Interface (GUI) and are much more platform dependent than other reported applications. In fact, while it is possible to run either Gnome or KDE on some other Unix-like distributions, it is not possible to run them on other platforms such as Microsoft Windows (which has its own GUI). This portrait is consistent with the data gathered, which suggests a strong usage of Microsoft Windows as the main Desktop operating system (see Fig. 4), on top of which, with varying degrees, FLOSS tools are installed. The appearance of Mozilla as the most used software is shown in the figure below.

The reason why there is no score amid the exclusive use of FLOSS on desktops, while there was a total of 5% and 6% of respondents declaring that they have FLOSS GUI distributions on their machines (as mentioned, are usually run on FLOSS OS, though they might be run also on some other Unix-like non-FLOSS OS) might be explained by the so called Dual Boot configuration. In development environments and amongst experts, quite often a single desktop machine is configured in a way that, when the power button is pressed, a program called Boot Loader opens and asks what operating system (and/or kernel) should be loaded. In this case, many operating systems can reside simultaneously on the same machine without the possibility of running contemporaneously (virtualisation is another issue).

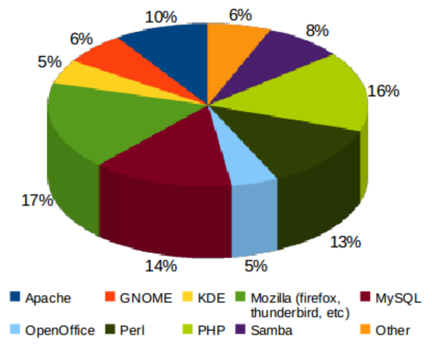

Figure 3: What kind of programs or applications are in use in your department?

That being said, such a configuration is relegated by a large extent to experimental uses (see Fig. 2), demonstrating that one of the two OS is not meant for productivity. In the present case, this most likely means that FLOSS might be installed but not used.

It is more difficult to explain the reason why only a very limited number of respondents (10%) declared to use Apache on their servers. Apache is a web server and is very widely distributed (official statistics of June 2010 report a total usage of more than 54% of the World Wide Web [32]). The reason for this might be found on a taxonomic level. In fact, Apache is not usually associated with FLOSS, especially with Gnu-Linux (it is released under the Apache licence, a FLOSS licence, but not GPL), and therefore a perception might exist where this type of tool does not pertain to the FLOSS family. Our questionnaire was purposely vague in asking what kind of FLOSS tools are in use, and may have resulted in the respondents discarded Apache if they do not believe it to be FLOSS. If this is the explanation of why our data do not mirror the general market situation, it is noteworthy that specifically trained IT departments are not aware of this misconception which is taxonomic in its origins, but very pragmatic in its consequences. Of course, it may simply be that the representative of the department did not know.

Our data also suggests that the use of OS in our analysis reflects the situation in the general market, where the dominance of Microsoft Windows (client side) is clear (world market data report 85% to 90% of MS Windows usage on clients [30][31]). In our enquiry, 48% of respondents declared that they use Microsoft products in one of its variants (98, XP, NT, Vista, 7, etc,). Gnu-Linux (in any of its variants: Debian, Red Hat/Fedora, Ubuntu, Slackware, Mandrivia, SuSe, etc) followed at 21%, then MacOS/X at 12%. In the case of Macintosh, the data closely mirrors the general data reported by the referenced statistics, however, the numbers regarding Windows and Linux do not accurately reflect the same data. In our data Windows achieves a 48% (contrasting with 85% to 90%) and Linux, a 21% (contrasting with 3%). Our data might suggest that there is wider use of the Gnu-Linux OS in the Canadian public sector; however, we must temper such an inference as our survey asked what types of OS are run on the (theoretically thousands of) clients managed by the respondents.

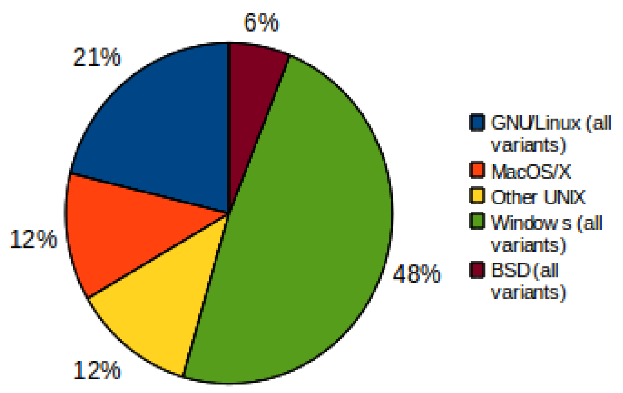

Figure 4: What kind of operating systems are installed in the department?

It is well known that FLOSS solutions, are not commonly used, as main operating systems on desktops. Nevertheless, an overwhelming majority of those interviewed agreed that a higher deployment of FLOSS would be beneficial to their department (65%). Less than one fifth of the respondents did not agree with this sentiment, it must be noted that a higher deployment does not equate to integral substitution. Out of the 65% of respondents in favour of a wider usage, only 11% would welcome a total substitution of their current software with FLOSS. Conversely, 78% would prefer a coexistence of proprietary and FLOSS.

It is interesting to note that that access to the source code is not the most important parameter to users that answered that they would welcome the use of FLOSS in their desktop systems (only 27% believe this). A far more important consideration was the price: 75% of respondents agreed that pure access to the source code (which includes the possibility to modify and redistribute it), not combined with the elimination of costs associated with licensing would render FLOSS unattractive to their departments.

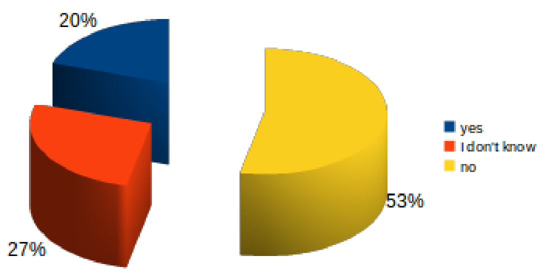

Figure 5: Is access to the source code an important factor for the department?

Regarding the more technical aspects, it was observed that a vast majority (85%) of interviewees acknowledged that FLOSS has higher customisation capabilities. Again, customisation is a characteristic that streams directly from the source code availability; however, the respondents (which were carefully identified in the IT departments) did not see such a connection. On the contrary, a majority of respondents believed that the main advantage of FLOSS is the fact that there are no licence fees. Another counter-intuitive result is connected with software reliability: 64% of the respondents believed that FLOSS is less reliable than non-FLOSS software. As discussed above, reliability may be considered an open issue, with strong advocates on both sides, supported by studies and data. Interestingly, among the surveyed category, there is a significant concentration of supporters of one specific view of the matter. More in keeping with the general perception were the respondents' views in terms of the ease for normal (i.e., non-technical) users in using FLOSS: 74% agreed that FLOSS is more difficult or complicated at first use. However, 80% did not see migration and training as an impediment towards a wider usage of FLOSS. As such if an adequate migration and training process is scheduled, the initial unease connected with FLOSS should be overcome.

The data reported so far give us a contradictory portrait of the perception of FLOSS in the Canadian public sector. We have seen how some of the positions held by the majority of the respondents are significantly diverging from general market trends. For example, in terms of reliability, FLOSS is believed to be much more reliable and accountable by large numbers. The same divergence is observable in more technical aspects, such as the availability of the source code. In the technical arena, this aspect (not only in FLOSS cases – think, for example, of beta-tester, premium user/developer of specific applications, specific important institutional customers such as homeland security departments, etc.) is considered essential for a great majority of the benefits we have identified above in this study. In our survey, however, respondents did not put much importance on the availability of the source code. Conversely, respondents believed the monetary aspect to be much more important. This is notable, considering that the respondents were IT departments of public administrations whose main objective is not to make profit but to offer a public service.

This does not mean that a public administration should not conform its activity simply to principles such as economisation, efficiency and rational usage of resources. On the contrary, it is exactly for these reasons that they should implement solutions that grant longer-term savings both monetary and in the possibility to re-utilise and scale-economise the (software) resources they use/produce. A fair and balanced administration of the public good is a science based on principles such as rationality, efficiency, accountability, and transparency. The tools analysed here, for the reasons explained in the relevant sections, are the most suited to meet both economic and social requirements of the management of public bodies.

A possible explanation for the contradictory feedbacks in our survey – impressions supported by the oral comments and further notes expressed during the interviews – is that even in the IT manager area the situation is strongly polarised or even ideological. On one side there are the majority who are supporters of one model, i.e., closed source software, who are prejudicially against any alternative model seen as a threat to "their model and to their jobs." Affirmations such as "we do not use any Open Source Software, we pay our licences!" or "we have internal guidelines not to use any Open Source software, so I had to remove also some amusement machines I had in my office" help to clarify why we have used such evocative wording.

In the middle there is a small category (approximately as large as the category supporting FLOSS) of IT departments who are undecided in which model is better; they perceive the pros and cons of both models, and who – most of the time – simply do their jobs "with the tools at their disposal". One 'Chief Technology Officer' said that the GoC is interested in FLOSS and encourages its use even though there is some uncertainty regarding the Intellectual Property issues and connected responsibilities.

There are also overt supporters of the FLOSS model; a minority who are strongly motivated and use FLOSS in their departments. These people are either working on the technical side (server, database management, programming, etc.) or in productivity workstations. These subjects are more sensitive to issues such as the availability of the source code, though do not delve deeply into the reasons why a public administration, more than a private corporation, should implement FLOSS solutions.

However, IT departments should be concerned mainly with technical decisions, while the more substantive ones should come from representatives of the decision-making bodies whose subjects are specifically appointed and trained to evaluate a great many differences in variables in choosing a fundamental instrument like the technological infrastructure of a public body. Such a simplified tri-partition of the respondents is particularly important because they (the respondent IT departments) have identified themselves as the decision-making subjects when purchasing new software in a good deal of cases (43%), while the financial department decides in only 17% of the cases.

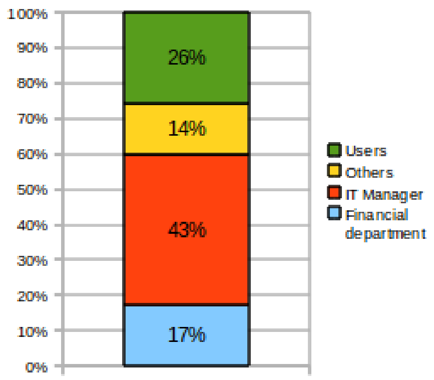

Figure 6: Whose opinion is decisive at the moment of purchasing new software?

VI. Recommendations and Conclusions

The GoC is not taking advantage of the many different features that an innovative model of software production and distribution, such as FLOSS, offers.

The landscape that emerges from the data reported here is not encouraging. FLOSS has proven to possess a long list of advantages in comparison to other software development and distribution models, especially in the public sector. The TCO is not the major concern, but together with the other FLOSS advantages exposed in this study, is an important one. Also in the TCO, FLOSS proved to be a strongly competitive and innovative model. However, easily discernible from our data is that the Canadian public sector completely lacks any coordination or guidelines in deciding which type of software tools to adopt in their departments. Even if guidelines existed, they are largely unattended, which paradoxically means that the Canadian governmental bodies distend their own rules.

Information Technology and, less frequently, financial departments decide which model to adopt, however, neither department are equipped or trained in making these decisions and therefore cannot take on the responsibility alone. As mentioned above, those departments are mainly concerned by financial or technical (customisation, stability, interoperability) issues – and it could not be otherwise. The problem is not so much what the Information Technology and financial departments believe but that in the majority of cases, they have the last word in deciding what to buy. Information Technology departments carry on a fundamentally important task and they have great experience. However, Information Technology departments cannot be left alone when making decisions regarding software use. Software is not a mere product but is a choice involving specific policy and political decisions that represent a specific set of values, public morality and ethics. Such political decisions need to be made by those whom have been elected who are ultimately responsible for the financial repercussions of software use in the public sector.

In short, there are many advantages that have strong economic value, in both the short and long term, that can only be eventualised by adopting FLOSS, with the technical and legal availability of the source code, and the possibility of its modification and redistribution. In addition to the monetary savings connected with the absence of licence fees, there are huge advantages that relate to the independence from software providers, the creation of a competitive market usually on a provincial, or regional level, transparency and accountability, the ease of customisation, digital inclusion and pluralism, and the further savings connected with scale economies amongst different public administrations [34]. The GoC should take full advantage of FLOSS in its technological infrastructure, because, as demonstrated, in many situations and at manly levels it would be beneficial to Canada. Currently it is not making full consideration of FLOSS.

Acknowledgment

The authors would like to thank SSHRC, and the Law Foundation of Ontario for their support; also Constance Keunhee Yoo and David William George Morrison for their research assistance.

References

[*] The current article represents an extended version of the follwoing paper: M. Perry and T. Margoni, '*FLOSS for the Canadian Public Sector: Open Democracy*', in Digital Society, 2010; ICDS '10. Fourth International Conference on, pp.294-300, 10-16 Feb. 2010

[1] R. Stallman, '*The Free Software Definition*', available at http://www.gnu.org/philosophy/free-sw.html; All the websites cited in this work have been last visited during January 2011.

[2] See 'Canadian Public Sector Contracts, Bids, and Tenders' web-site at http://www.merx.com.

[3] See for example R. Goldman and R. Gabriel, '*Innovation Happens Elsewhere - Open Source as Business Strategy*', Morgan Kaufman - Elsevier, San Francisco, 2005.

[4] R. Stallman, *'Why "Open Source" misses the point of Free Software'*, available at http://www.gnu.org/philosophy/open-source-misses-the-point.html.

[5] M. Tiemann, '*History of the OSI*', available at http://www.opensource.org/history.

[6] See 'Freshmeat GPL tagged projects' available at http://freshmeat.net/tags/gnu-general-public-license-gpl.

[7] B. Schneier, *Open Source and Security*, in Crypto-Gram Newsletter, September 15, 1999.

[8] For a deep analysis of legal issues surrounding FLOSS see L. Guibault and O. van Daalen, '*Unravelling the Myth around Open Source Licences*', ITeR, The Hague, 2006.

[9] H. Kaminski; L. Kari; and M. Perry, '*Who counts your votes? [VEV electronic voting system],*' e-Technology, e-Commerce and e-Service, 2005. EEE '05, Proceedings. The 2005 IEEE International Conference pp. 598-603, 29 March-1 April 2005.

[10] See arts. 68 and 69 *Codice Amministrazione Digitale*, Legislative Decree 7 March 2006, n. 82, (as amended).

[11] See for example the 'Microsoft Open Source Interoperability Initiative' at http://www.microsoft.com/interop/.

[12] See the GPL/Plug-ins FAQ at http://www.gnu.org/licenses/gpl-faq.htmlGPLAndPlugins.

[13] See Prism Hospital Software Inc. v. Hospital Medical Records Institute [1994], 97 B.C.L.R. (2d) 201, [1994] 10 W.W.R. 305, 57 C.P.R. (3d) 129, 18 B.L.R. (2d) 1.

[14] See Apple Computer Inc. v. Mackintosh Computers Ltd., [1990] 2 S.C.R. 209, 110 N.R. 66, 30 C.P.R. (3d) 257, 71 D.L.R. (4th) 95, 36 F.T.R. 159.

[15] D. Vaver, '*Translation and Copyright: a Canadian focus*'; in E.I.P.R., 1994, 16(4), 159-166, at 160.

[16] See the 'GPL only modules' thread at lkml.org, availabe at http://lkml.org/lkml/2006/12/17/79.

[17] See the Mozilla Public Licence FAQ at http://www.mozilla.org/MPL/mpl-faq.html.

[18] See the Common (now Eclipse) Public Licence available at http://www.eclipse.org/legal/cpl-v10.html.

[19] B. Scott, '*Lock in Software*', Open Source Law Publications, 2003.

[20] B. Perens, '*Open Standard, Principles and Practice*', available at http://perens.com/OpenStandards/Definition.html.

[21] D.A. Wheeler, '*Secure programming for Linux and Unix HowTo*', 2003, available under the GFDL license at http://www.dwheeler.com/secure-programs.

[22] M. Perry and B. Fitzgerald, '*FLOSS as Democratic Principle*', in International Journal of Technology, Knowledge, and Society, vol. II, 3, 2006, pp. 156 – 164.

[23] K. Gerloff, '*Declaration of Independence: the LiMux project in Munich*', Open Source Observatory and Repository (OSOR), European Commission's IDABC project.

[24] Department of Finance and Administration '*A guide to Open Source Software for Australian government agencies*', Australian Government Information Management Office, 2005.

[25] M. Perry and T. Margoni, '*Floss for the Canadian public sector: Open Democracy*', in Digital Society 2010, 2010, pp. 294

[26] K. Wong, '*Free/Open Source Software – Government Policy*', UNDP – Asia-Pacific Development Information Program, Elsevier, New Dehli, 2004.

[27] The Swedish Agency for Public Management '*Free and Open Source Software – a feasibility study*', Stockholm, 2003.

[28] Source: *Free/Libre and Open Source Software: Survey and Study*, International Institute of Infonomics, University of Maastricht, The Netherlands, June 2002, available at http://www.flossproject.org.

[29] See '*FlossPols Government Survey Report*', Deliverable D3, Maastricht, August 25, 2005 - MERIT, University of Maastricht, available at http://flosspols.org.

[30] See 'Operating System Market Share' at http://marketshare.hitslink.com/operating-system-market-share.aspx?qprid=8.

[31] See 'Global Web Stats' at http://www.w3counter.com/globalstats.php.

[32] See 'January 2011 Web Server Survey' available at http://news.netcraft.com/archives/category/web-server-survey.

[33] See LiMux web-page project at http://www.muenchen.de/Rathaus/dir/limux/english/147197/index.html.

[34] Cenatic, '*Software de fuentes abiertas para el desarrollo de la administracion Publica Espanola - Una vision global*', Observatorio Nacional de Software de fuentes abiertas, Badajoz, 2008.

[35] B. Boyle, '*Open Source Software*', Minister of State Service of New Zealand, New Zealand, 2003.

[36] State Service Commission, '*Guide to legal issues in using Open Source Software v2*', New Zealand Government, 2006.

[37] Y. Benkler, '*The wealth of networks – how social productions transforms markets and freedom*', Yale University Press, 2006.

[38] J. Dickson, '*Use of open source: licenses and issues*', in e-Commerce Law and Policy, March 2009, pp. 8.

[39] B. Fitzgerald and N. Suzor, '*Legal issues for the use of free and open source software in government*', in Melbourne University Law Review, 29, 2005, pp. 412.