Telematics and Informatics 30 (2013) 100–110

Contents lists available at SciVerse ScienceDirect

Telematics and Informatics

journal homepage: www.elsevier.com/locate/tele

Methodology for Public Administrators for selecting between open source and proprietary software ☆

Christos Bouras *, Vasileios Kokkinos, Georgia Tseliou

Computer Technology Institute & Press "Diophantus", Patras, Greece
Computer Engineering and Informatics Department, University of Patras, Greece

ARTICLE INFO

Article history:
Received 10 November 2011
Received in revised form 2 March 2012
Accepted 7 March 2012
Available online 21 March 2012

Keywords:
Open source software
Proprietary software
Public Administration guidelines

ABSTRACT

The public sector needs to change over to communicating digitally. This development makes great demands both on work processes in the public sector and on the Information Technology systems, on which e-government is based. From the economic perspective, the change-over poses great challenges, as huge investments will have to be made in Information Technology in the public sector. It is therefore natural, in connection with these investments, for a detailed assessment to be made of what forms of technology it is anticipated to be used, and who controls the development and ownership of this technology. The question is: to what extent Free and Open Source Software can supplement or completely replace proprietary software? This work constitutes a review of literature on pre-existing comparative studies regarding the technical, social, economic and organizational factors on Free and Open Source Software usage. Furthermore, this work includes guidelines that Public Administrations should follow for the selection between open source and proprietary software. Our goal is to help public stakeholders understand the technical/social/economic/organizational environment and therefore reach informed decisions when selecting the appropriate software. The manuscript can also be useful for Free and Open Source Software developers, users and communities who are either directly or indirectly involved in the software market.

1. Introduction

Software (SW) can be shortly defined as the executable code that controls computer behavior and operations. The term is used, however, to describe a wide range of programming languages, applications, procedures and all related documentation resources. SW also refers to a full cycle of processes from basic architecture to development, packaging and distributing. It is responsible for controlling, integrating, and managing the individual hardware components of a computer system in order for other software and the users of the system to be able to see it as a functional unit without having to be concerned with the low-level details of the computational system.

Although there are different definitions of Free and Open Source Software (FOSS), there are some basic principles on which FOSS relies. These refer to the freedom to run a software program for any purpose, to study and modify a software program by accessing its source code and to distribute copies of a software program, whether modified or not. Additional

☆ Work supported by the ERDF – EU National funded Interregional Cooperation Programme (INTERREG IVC) under contract number 0918R2 (project: OSEPA – Open Source software usage by European Public Administrations).

* Corresponding author at: Computer Technology Institute & Press "Diophantus", Greece, N. Kazantzaki, GR-26504 Patras, Greece. Tel.: +30 2610 996951; fax: +30 2610 969016.

E-mail address: bouras@cti.gr (C. Bouras).

0736-5853/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.tele.2012.03.001

prerequisites for FOSS-programs include: no discrimination against persons, groups or fields of endeavor and distributable, technology-neutral licenses that are not specific to a product or restrict other software. These freedoms and principles are defined by the Free Software Foundation (http://www.gnu.org/philosophy/free-sw.html) and the Open Source Initiative (http://www.opensource.org/osd.html).

FOSS has become very popular in the last few years and is advancing at a speed unknown outside the world of Information Technology (IT). Just a few years ago, FOSS was not regarded as a popular solution compared to the solutions provided by the giants of IT. Today, FOSS has become an area of business – an alternative, and therefore a competitor, to proprietary software.

This interest has also spread to the world of politics. This occurs due to different parameters: FOSS is now increasingly used for commercial purposes, it is characterized as independent from software producers, it is opposed to the creation of monopolies and it is characterized by a "free-of-charge principle". Furthermore, FOSS has had a great impact on the political agenda, nationally and internationally.

FOSS-solutions have several advantages and disadvantages compared to proprietary SW solutions. This manuscript analyses the main factors that affect FOSS-use and adoption by Public Administrators (PAs). Moreover, this work describes some basic and important guidelines that should be followed for the evaluation and adoption of any software. The basic steps for evaluating all programs, both FOSS and proprietary SW, are essentially the same. However, the way that these steps are performed in an evaluation process is different for FOSS-programs than for proprietary ones. A key difference for evaluation is that the information available for FOSS-programs is usually different than for proprietary programs.

Indeed, most FOSS-programs have a great deal of publicly available information that is not available for proprietary programs: the program's source code, analysis by others of the program design, discussions between developers about its design and future directions, discussions between users and developers on how well it is working (or not), and so on. An even more fundamental difference between FOSS and proprietary programs is that FOSS-programs can be changed and be redistributed by customers. This difference affects many parameters, such as support options, flexibility, customizability and costs. Proprietary programs generally do not give the user the right to view, modify, and redistribute a program, and it would not make sense to ignore these vital differences. Some administrators may decide that they wish to only use FOSS-programs. However, even in this case, the user still needs to be able to evaluate FOSS-programs, because there is always the need to know how well a given program meets the user's needs, and there are often competing FOSS-programs.

The rest of this manuscript is structured as follows: Section 2 describes in detail the work related to our study and Section 3 describes the factors that affect FOSS-use and adoption by PAs. A detailed list of guidelines for selecting between FOSS and proprietary software is provided in Section 4. Finally in Sections 5 and 6 our conclusions and some proposals for future work are drawn up.

2. Review of literature

This section constitutes a review of literature on pre-existing comparative studies and surveys regarding the technical, organizational, economic and social factors of FOSS-usage. The surveys were executed in various regions or sectors where FOSS is applied.

In Floss Final Report (2002), a survey that is intended to yield information about FOSS-use in several countries of the European Union is presented. Due to budgetary restrictions, interviews could only be conducted for a limited number of countries (Germany, Sweden and the UK). One of the results of this survey is that FOSS-usage rates and technological issues not only differ by country, but also within countries. Another survey (presented in Olsoon et al. (2010)) that was conducted in Sweden answers the question of how common the usage of FOSS is, by informing the public that 50% of the local authorities use FOSS, mainly in operating systems. Also, this work presents examples that concern functionality, support and maintenance of FOSS-issues.

Taking in mind the analysis in Moolman (2011), it must be noted that technological factors affect FOSS on a large scale. People that support the adoption of FOSS believe that FOSS shows more stable behavior than proprietary software. The authors of Dedrick and West, 2007 claim that the use of FOSS in organizations still has to be motivated on utilitarian grounds. Technological factors that show a relevance to FOSS-adoption include maturity, performance, stability, usability, security and quality of support.

As stated in Moolman (2011), previous experience with FOSS plays a significant role in the selection of such kind of software. Usually, organizations with little or no experience in FOSS are better off choosing software. This happens due to the fact that mature FOSS-solutions supported by commercial companies and universities generally present a lower risk as they have been adopted by many organizations and documentation as well as support is available. It is quite interesting to observe that several FOSS-projects considered immature when measured with maturity models are in fact mature enough for adoption, given that the adopting organization has some FOSS-experience (Ven et al., 2008).

The same authors mention the maturity of the organization dealing with FOSS in James and Van Belle (2008). Their measure of maturity also takes into account the intended application within the organization, availability of support and the maturity of the development community behind the software. They highlight maturity factors that are organization-centric, solution-centric or external entity-centric. They concluded that the maturity of the solution under review is dependent on its intended application within the organization.

Software maturity is a decision factor that depends on the environment in which the software is used (James and Van Belle, 2008; Holck et al., 2005). Reliability is an important aspect of software maturity and mature software is also seen

as reliable. Reliability comparisons between FOSS and proprietary software are almost futile as both software types cover a range of software from extremely stable to rather unstable.

There are several publications regarding organizational issues about FOSS. The survey presented in Olsoon et al. (2010) pointed out, that there is a great need for support in procurement and utilization of FOSS. Moreover, the survey in Rentocch and Tartary (2007) presents some obstacles to the adoption of Information and Communications Technology (ICT) by the Public Administrators (PAs). The survey noted immediately that there are differences among municipalities with different intensity degree. Municipalities with a high intensity of FOSS-adoption, rate the low flexibility of suppliers and the low interoperability of applications as the main obstacles to a correct implementation of the ICTs. For the two other groups, namely moderate intensity and no intensity, the main obstacles are the low number of employees and high costs.

An organizational factor that affects the adoption of FOSS is the lack of awareness that can be remedied by having FOSS advocates and boundary spanners working in an organization. Definitely boundary spanners are effective in connecting organizations to new technologies and provide the skills and knowledge needed for successful adoption (Ven and Verelst, 2009). FOSS-champions successfully influence adoption decisions from within an organization, reducing some of the individual uncertainty and fear (Morgan and Finnegan, 2007). The amount of influence FOSS-champions have within an organization is determined by the institutional limitations in the organization and their position within the organization (Holck et al., 2005). There are many economic factors that can be considered in social environments and affect the adoption of FOSS. A business benefit that can be considered is cost reduction in relation to technical benefits and drawbacks of FOSS-adoption (Morgan and Finnegan, 2007).

It is interesting to observe that cost as a factor in FOSS-adoption decisions depends on an objective measurement. The authors in Richter et al. (2009) found that for many companies, FOSS-adoption is centered on value creation. The advantage however comes not only from costs which are saved but it also benefits in terms of reliability, flexibility and a higher degree of innovation capability.

The purpose of Danish Board of Technology (2002) is to illustrate the socio-economic differences between the use of FOSS and proprietary software in PAs in Denmark. The conducted socio-economic analysis assesses the total loss that follows from decisions taken against the background of limited information and imperfect market competition. The business case of FOSS-adoption is driven by lower costs, but it is also dependent on the application area, company size and price elasticity in the market. Application area and adoption scale is important as it might be prohibitively expensive to make a company-wide switch from one platform to another (Holck et al., 2005). The level of strategic importance of software to the business also plays a role in adoption decisions. Software with low strategic importance and high price sensitivity tend to be better candidates for FOSS-adoption (Kwan and West, 2005).

Developing countries, in general, adopt FOSS due to cost advantages. The effect of software license fees are more pronounced in developing countries as they make up a larger part of the total system cost when taking into account hardware and software. Lower labor costs mean that license fees constitute a bigger percentage of IT costs (Paudel et al., 2010). One interesting example occurs in the German public sector where low cost is one of the main drivers of FOSS-adoption. The German foreign office started migrating to FOSS in 2002 and by 2005 it was the "cheapest" ministry in German government in terms of IT expenditure. In Brazil, the government uses FOSS to save on license fees, keeping money that was previously paid to foreign vendors inside the country (Richter et al., 2009). Through collaboration with local industries costs can be minimized and national competitiveness in software industries can be improved (Hwang, 2005).

Considering a survey that was conducted in the UK (presented in Public Sector (2009)) almost two-thirds of those surveyed believe that the benefits of open source generally outweigh its drawbacks. However, the general consensus is that local government fails to give sufficient consideration to open source in software procurements. The research finds that open source use in local government will, overall, only keep increasing. The majority view (42%) is that local authorities will increase their use of FOSS over the next 3 years. Around a third of those surveyed expect current levels of adoption to remain unchanged during this period. This highlights a significant degree of uncertainty among sections of local government over plans for future adoption.

The authors in James and Van Belle (2008) found that in rural areas, even in migrations where all other factors have been taken into account, the social interaction of people involved could lead to a perfect migration failing. Most initial FOSS initiatives in the Asia–Pacific region have dealt with localizing FOSS. There is a variety of social factors that make FOSS really attractive for use in public administrations. There are many examples that confirm this assertion. When FOSS and government are mentioned in the same context, it is usually in relation to public sector use of software. Sometimes, it is related to public sector policies for the promotion of FOSS. But there is also an increasing number of projects producing customized software for public administrations (Wong and Sayo, 2004).

IT professionals from four continents have collaborated with the National Open Source Observatory (CENATIC), and their opinions and suggestions have served as the basis for the conclusions set out in the dossier presented in CENATIC. In terms of the criteria for adopting FOSS, the dossier concludes that the administrations are influenced by criteria such as vendor independence and flexibility, as well as open standards and open development process. However, the public administrations are less easily persuaded by criteria such as faster procurement, best-of-breed solutions and political decisions and initiatives.

In brief, we can observe that there are several publications that take into account the variety of issues that concern FOSS. The main technological issues that were presented in the literature concern functionality and support. Also, important aspects are software maintenance, management and performance. According, to the related work, IT administrators claim that organizational structure is important. IT infrastructure is often decentralized so that each department has its own IT

person. Moreover, although the low price of FOSS-products is the primary factor for using these products, the literature has introduced other economic perspectives that are related to FOSS. Finally, social issues such as knowledge sharing, satisfaction of achieving something valuable, learning and improving personal skills and legal aspects of software play a significant role in FOSS-adoption and use.

3. Factors affecting FOSS usage and adoption

In this section, we describe all the factors that affect the FOSS-usage and adoption by PAs. Table 1 summarizes all the factors that were found in the surveys presented in Section 2 and indicates which survey contains each one of them. The paragraphs that follow present and analyze the most significant factors.

3.1. Technological factors

3.1.1. Functionality

It refers to the degree that a program integrates and it is compatible with existing components. Functionality also means if there are relevant standards and if the program supports them and what hardware, operating systems and related programs are required (Wheeler, 2011).

Table 1
Factors affecting FOSS usage and adoption.

Factor		Surveys					
		FLOSS	Sambruk	Danish Board of Technology	EROSS	Public Sector Forum	CENATIC
Technological	Functionality	X	X				
	Support	X	X				
	Maintenance	X	X				
	Management	X					
	Longevity	X					
	Reliability	X					
	Availability	X					X
	Security	X					
	Performance	X					
	Usability	X					
	Interoperability	X					X
	Integration	X					X
	Trialability	X					
	Data Migration	X					
Organizational	Ability to find the right staff and competencies		X		X	X	
	Lack of awareness		X		X	X	
	Training issues		X		X	X	X
	Resistance to change		X		X	X	
	Strong leadership		X		X	X	X
	Management support		X		X	X	
	Availability of in-house skills and knowledge		X		X	X	
	Real world experience		X		X	X	
	Interoperability of applications		X		X	X	
Cost/economic	Labor costs			X	X	X	X
	Control the costs of software licensing and upgrades		X	X	X	X	X
	Control and increase the access to intellectual properties			X	X	X	X
	Promote software use in the public sectors			X	X	X	
Social	Knowledge sharing			X			
	Satisfaction of achieving something valuable			X			X
	Professional reputation and recognition among peers			X			
	Learning/Improving personal skills			X			X
	Legal aspects		X	X		X	
	Sense of belonging to the community			X			
	Enjoyment of developing			X			X
	Sharing knowledge			X			
	Improving products			X			
	Freedom in developing SW			X			
	Learning and developing new skills			X			X
	Sense of belonging to the community			X			X

3.1.2. Support

The term "support" covers several areas. For example it refers to training users on how to use the product, installing the product, helping users who have specific problems when trying to use a working product. This includes product documentation (user documentation, reference guides, and any other source of information), warranty or indemnification. One major difference between FOSS and proprietary programs is how support is handled. Fundamentally, FOSS-program users have several choices such as choosing a traditional commercial support model, where they pay someone to provide support, providing support in-house and depending on the development and user community for support. These choices apply to each of the various tasks (training, installing, answering questions, fixing, adding new capabilities), and the answers can even be different for the different tasks. Unlike proprietary support (which is usually only provided by the proprietary vendor), there may be several competing companies offering support (Wheeler, 2011).

3.1.3. Maintenance/management/longevity

FOSS-maintenance options are essentially the same as those for support, and in reality maintenance and support are not completely separate (Wheeler, 2011). Few useful programs are completely static. Needs change, new uses are continuously created, and no program of any kind is perfect. It is important that a program is being maintained, and that it will be maintained far into the future. Of course, predicting the future is very difficult. However, if a program is being actively maintained, it is far more likely that the program will be useful in the future. The human team which is in charge of the maintenance process should also be responsible for the characterization of the aging of a system. A maintainer who has some knowledge about a system will maintain it better than somebody with no experience with it. Several models for the measurement of software maintenance and management exist, such as the hierarchical models presented in Oman and Hagemeister (1992). The longevity factor is related to the time period in which a project has existed combined with a healthy level of activity and it is a measure of the chance of survival of a project.

3.1.4. Reliability/availability

Reliability measures how often the program works and produces the appropriate answers. A quite similar measure is availability. Reliability is difficult to measure, and strongly depends on how the program is used (Wheeler, 2011). Problem reports are not necessarily a sign of poor reliability – people often complain about highly reliable programs, because their high reliability often leads both customers and engineers to extremely high expectations.

3.1.5. Security

Evaluating a product's security is complicated, in part because different uses and environments often impose different security requirements on the same type of product. One step toward solving this problem is the identification of security requirements (Wheeler, 2011). FOSS, offers security due to the fact that when a programmer creates a program and decides to open the source code to the public, other programmers can take a look at the code and communicate with the original programmer about what they feel should be the right way to approach, for example, a specific part of the program. This process creates strong best-practices and forces FOSS-programmers to follow standards.

3.1.6. Performance

Many project websites include performance data. It is worth mentioning that some FOSS-projects, unsurprisingly, only present the most positive performance data near their front pages, so this may portray the whole picture. Project mailing lists may include more detailed performance information (Wheeler, 2011).

3.1.7. Usability

Usability measures the quality of the human–machine interface for its intended user. A highly usable program is easier to learn and easier to use. Some programs (typically computer libraries) are intended only for use by other programs and not directly by users. In that case, it will typically have an Application Programmer Interface (API) and someone should measure how easily programmers can use it. Generally, an API should make the simple things simple and the hard things possible. One way to get a measure of this is to look for sample fragments of code that use the API, in order to see how easy it is to use.

3.1.8. Flexibility/customizability

Flexibility and customizability are two highly interrelated attributes which measure how well a program can be used to handle unusual circumstances that it was not originally designed for and how well someone can customize the product to fit into his/her specific environment.

3.1.9. Interoperability/integration

Before selecting a software product, it is necessary to ensure that it will work with the other products that the interested user already uses or plans to use. Moreover, it is advised products to use standards – this way the best product can be chosen (instead of being locked into one vendor's product), and change later. The user/organization does not depend on a given vendor and thus mitigates the software investment risk of a vendor going bankrupt or being acquired by a competitor that discontinues the software. FOSS-products typically implement relevant standards, simply because there is usually no good

economic reason not to Wheeler (2011). True interoperability relies on open standards, which should make it possible to swap out individual components for better or competing offerings by someone else. If true interoperability is what an organization is looking for, then the organization should definitely choose open source, since it implements open data standards by default.

3.1.10. Trialability

Trialability is one of the factors in the classic Diffusion of Innovations (DOI) theory and refers to the ability to try out a new innovation on a limited basis before making a decision on whether to adopt the innovation or not. Trialability of an innovation is hypothesized to be positively related to the adoption of that innovation. FOSS is easier to try out than commercial software, because a full version of the software can be freely downloaded from the Internet (Oman and Hagemeister, 1992). It is very important to be able to try a software product before using it in a production environment.

3.1.11. Technical issues for the migration to FOSS solutions

Some of the problems related to the migration to FOSS-solutions are listed below (Varghese, 2003):

- possible need for extensive migration,
- could lead to higher demands for in-house competence and maintenance within the agency or authority itself,
- could be difficult finding the right product,
- possible interoperability problems with proprietary software,
- fewer available consultant and support services on the market at present time.

3.2. Organizational factors

Despite the attention that FOSS has received, relatively little is known about the factors which influence the decision of an organization on whether to adopt FOSS or not. Although much anecdotal evidence has been published on this topic in practitioner literature, these claims have been insufficiently validated. Although some qualitative studies on the organizational adoption of FOSS have been conducted, empirical support based on a large sample is missing. According to Moolman (2011) some other organizational factors that affect FOSS usage are the following:

- ability to find the right staff and competencies needed to adopt FOSS,
- social interaction of people involved in FOSS,
- lack of awareness,
- training issues,
- resistance to change,
- strong leadership,
- management support of FOSS,
- availability of in-house FOSS skills and knowledge,
- lack of real world experience in FOSS adoption.

IT administrators said organizational structure is important. IT infrastructure is often decentralized so that each department has its own IT person. For example, officials in three German cities stressed that a change as fundamental as migrating to FOSS is easiest with a centralized IT department. Based on their experience with migration, the directors reported that a decentralized IT structure creates cultural and structural barriers in the organization that make it difficult to adopt a government wide strategy (CIO-Zone).

In Munich, for example, before migration to FOSS, IT was highly decentralized. More than 850 IT professionals were scattered across 17 departments. The departments did not resist change. Instead, when migration to FOSS was proposed, the city departments were reluctant to give up what they perceived as their IT professional(s) or expertise. This significantly slowed the migration process since migrating to FOSS required taking stock of the government's entire IT infrastructure, identifying FOSS-alternatives, and then standardizing the government's operating systems and software. Such a change is made easier by a centralized IT structure, regardless of the city's organizational culture. IT directors in all three cities noted that a centralized structure improved migration to FOSS (CIO-Zone).

3.3. Cost factors

There are many economic factors that can be considered in social environments and affect the adoption of FOSS. A benefit that can be considered is cost reduction in relation to technical benefits and drawbacks of FOSS-adoption (Morgan and Finnegan, 2007).

The business case for FOSS-adoption is driven by lower costs, but is also dependent on the application area and the organization size. Application area and adoption scale is important as it might be prohibitively expensive to make a switch from one platform to another (Holck et al., 2005). The level of strategic importance of software to the organization also plays a role

in adoption decisions. Software with low strategic importance and high price sensitivity tend to be better candidates for FOSS-adoption (Danish Board of Technology, 2002).

The importance of cost in FOSS-adoption decisions is dependent on the adoption scale. In Belgian firms, for small scale adoptions, lower cost played a significant role as it enabled experimentation with a limited budget (Holck et al., 2005). In large scale adoption, proprietary software license discounts become a barrier to FOSS-adoption compared to proprietary software.

Although the low price of FOSS-products is the primary factor for using these products, there are also other economic perspectives, not only in using FOSS but also in developing products. Four economic incentives for the adoption of FOSS-software and support its development by governments are the following:

- control the costs of software licensing and upgrades,
- control and increase the access to intellectual properties,
- reduce the reliance on proprietary software,
- promote software use in the public sectors.

3.4. Social factors

Community identification, self satisfaction, and fulfillment that arise from writing programs are considered as the motivation of FOSS developers, since their desire is to fulfill their personal needs (Hars and Ou, 2000). Internal motivation factors are summarized as follows:

- knowledge sharing,
- satisfaction of achieving something valuable,
- professional reputation and recognition among peers,
- learning and improving personal skills,
- group problem solving,
- challenge proprietary software,
- sense of belonging to the community,
- enjoyment of developing projects.

External factors show that the major reasons of developers' participation in FOSS software development are the following:

- learning and developing new skills,
- sharing knowledge,
- improving products,
- freedom in developing software.

It is noteworthy that the literature shows that knowledge sharing among participants is a key motivator that can be used in technology transfer. Some governments and other organizations have specific policies preferring FOSS-programs over proprietary programs. This is rare; what is unclear is whether or not this is a trend. Some governments are reluctant to store official records in the proprietary formats of proprietary software vendors because they believe the software's transparency increases security because security problems can be quickly exposed and fixed, the software can also be tailored to the user's specific needs, upgrades happen at a pace chosen by the user (not the vendor), and this move tends to benefit numerous small, local technology firms (Wheeler, 2011).

Localization is one of the areas where FOSS shines because of its open nature. Users are able to modify FOSS to suit the unique requirements of a particular cultural region, regardless of economic size. All that is necessary is the technical capability within a small number of individuals to create a minimal localized version of any FOSS. While the construction of a completely localized software platform is no small feat, it is at least possible.

In Section 3 the factors that affect the FOSS-usage and adoption by PAs were analyzed. The factors were distinguished into four categories: technological, organizational, cost and social factors. Based on this analysis, Section 4 proposes some guidelines that should be followed by PAs in order to evaluate these factors and select the most appropriate software solution (FOSS or proprietary). Before the evaluation and selection process, we have added some "logical" steps that a PA should follow in order to reach the optimal solution. These steps have been identified by the OSEPA consortium (INTERREG IVC project under contract number 0918R2) and do not rely on literature review. In detail, these preliminary basic steps are the "Formation of a special group of experts" and the "Identification of potential software solutions".

4. Guidelines for selecting among FOSS and proprietary software

Based on the previous analysis, FOSS-solutions have several advantages and disadvantages compared to proprietary SW solutions. In this section, we describe some basic and important guidelines that should be followed by organizations or PAs

for the adoption of any software. The basic steps for evaluating all programs, both FOSS and proprietary SW, are essentially the same. However, the way that these steps are performed in an evaluation process is different for FOSS-programs than for proprietary ones. A key difference for evaluation is that the information available for FOSS-programs is usually different than for proprietary programs. In Fig. 1 we present the steps that should be followed for selecting among FOSS and proprietary software. The paragraphs that follow present these steps in detail.

4.1. Formation of a special group of experts

Before the software search begins, it is necessary to form a search group of experts. This group should consist of the computer department and various department heads. This kind of approach has worked very well for many companies and public administrations (PAs). By pairing the computer department that specializes in technology with the heads of departments who know the business needs, the interested company or PA develops a very strong software search team.

The most successful installations have been with companies that had this kind of committee, in which the computer department becomes the liaison between the users and the software implementation team translating technology to their requirements (Floss Final Report, 2002).

4.2. Identification of potential software solutions

A combination of techniques should be used in order to make sure that something important is not neglected. An obvious way is for the interested user to make a questionnaire, whether other users (organizations or PAs) also need or have used such a program. If they have experience with it, they should ask for their critique; this will be useful as input for the next step, obtaining reviews.

Moreover, it is necessary to examine lists of programs, including any list of ''generally recognized as mature'' or ''generally recognized as safe'' programs. Some products are so well-known that it would be a terrible mistake to not consider them. It is advised that the interested user asks only a few of the most relevant lists. Also general systems can be used to make requests, such as Google answers, where someone pays a fee to get an answer. The search group of experts is more qualified to make a detailed search.

4.3. Study of existing reviews

After the identification of options, it is necessary to study all the existing evaluations about the alternatives. It is far more efficient to first learn about a program's strengths and weaknesses from a few reviews than to try to discern that information just from project websites. It is critical that many evaluations are biased or not particularly relevant to any circumstance. An important though indirect ''review'' of a product is the product's popularity, also known as market share.

Generally, a user should always try to include the most popular products in any evaluation. Products with large market share are likely to be sufficient for many needs, are often easier to support and interoperate, and so on. Developers do not want their work wasted, so they will want to work with projects perceived to be successful. Conversely, a product rapidly losing market share has a greater risk, because presumably people are leaving it for a reason.

4.4. Definition of technical areas and required components

It is very important, in any software selection or migration project, to have a clear view of the technical areas (server, client, and network) and software components (both open source and proprietary) that are required for installation and deployment. Server-based systems, for example, require pre-existing web or application servers and more advanced installation and configuration processes. Some applications also require a parallel deployment or co-existence of both open source and proprietary components that should be carefully taken into account in order to avoid compatibility failures.

4.5. Comparison of the leading programs' attributes to specific needs

There are some attributes that should be taken into consideration as far as the choice between FOSS and proprietary SW is concerned. Important attributes include the following: functionality, cost estimation (initial license fees, license upgrade fees, installation costs, staffing costs, support/maintenance costs, indirect costs such as training and transition costs), market share, support, trialability, maintenance/longevity, compatibility, reliability/availability, security, scalability, performance, usability, flexibility/customizability, interoperability, legal/license issues, local policies, possible data migration, organizational factors, human factors (e.g. resistance of personnel to change SW, maturity of the personnel, etc.), and other environmental and social factors. A detailed analysis of the attributes/factors that affect the FOSS-usage and adoption and that should be considered for reaching the optimal SW solution is presented in Section 3 and synopsized in Table 1.

The benefits, drawbacks, and risks of using a program can be determined from examining these attributes. The attributes are of course the same as with proprietary software, but the way that a user should evaluate them when it comes to FOSS or proprietary SW is often different. In particular, because the FOSS project and code is completely exposed to the world, the user must take advantage of this information during evaluation.

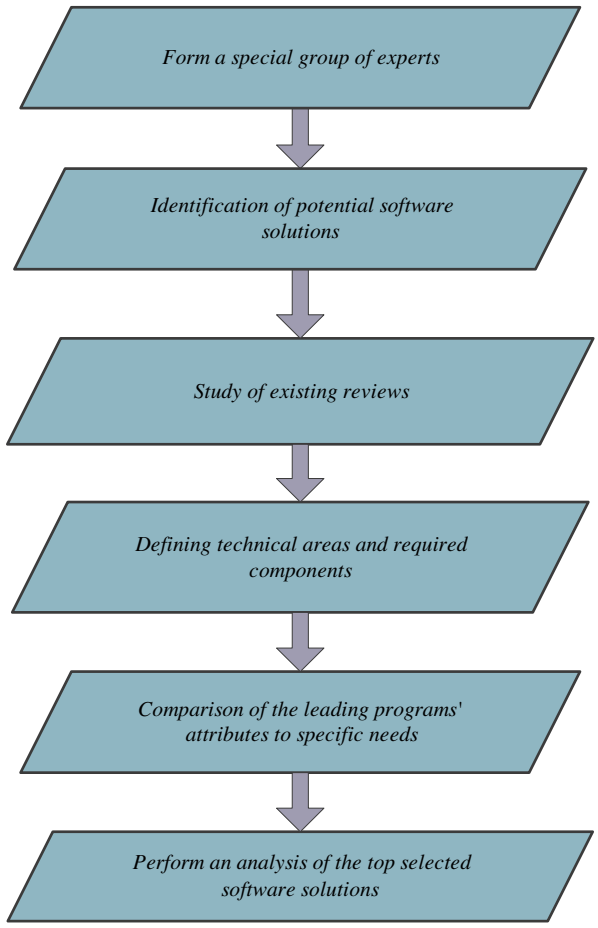

Fig. 1. Guidelines for selecting among Open Source and Proprietary Software.

4.6. Performance of an analysis of the top selected software solutions

After the evaluation, the organization picks the top candidates, and performs a more detailed analysis of them. This step is, for the most part, done in the same way for both proprietary and FOSS-programs. The important attributes to consider are the same as in the previous step.

More effort is spent by actually trying things out instead of quickly reading the available literature. For example, to see what functionality a program provides, a user would run it and try out the functionality that he/she is interested in using (e.g., if the user is concerned about interoperability, he/she will acquire some sample same files or systems and see how well it works). A user should always carefully identify the version number of the program, because the description of the first version may not be the same as in a later one. This is particularly important for FOSS programs, because many FOSS-programs undergo rapid improvement.

A more important difference is that in FOSS there are sources of information about a program that may not be available for proprietary software. In particular, a user can also have a software professional examine the program's design documentation, source code, and other related materials. The conducted analysis can be categorized in:

- Analysis for Adding Functionality.
- Analysis of Software Security.

Once a decision has been made, it is time to begin the process to install the new program.

A successful case that followed the procedure described in this section is presented in Salmivalli and Nissila (2004), where the European open source health care project SPIRIT was researched and analyzed. The basic steps of the methodology were the following:

- creation of a community of collaborative developers, users, and policy makers,
- research among five potential open source solutions,

- policy makers in healthcare IT from most European countries were approached on an individual basis or in particular conferences and meetings,
- creation of a virtual meeting place,
- increased commercial exploitation of selected open source resources.

5. Conclusions

In this work, a detailed list of guidelines for selecting among open source and proprietary software is presented. Concluding this work, it can be anticipated that FOSS will be more widely used over the next few years, as a smaller proportion of those organizations or PAs that do not use FOSS is expected to do so in the next 2 years, while a larger proportion of the organizations that already use FOSS is expected to increase their use of it. The decisive reasons why FOSS is used or not mainly relate to economic savings, awareness of FOSS, compatibility, the development of programs and user-friendliness.

The use, adoption and integration of FOSS in the IT infrastructures of European governments and public administrations has not always followed the same pace or moved towards the same direction. Legal and institutional frameworks, social, economic and technological aspects are some of the differentiating factors that explain gaps or divides between regions and countries on the awareness and penetration level of open source. Some countries are leading the way of open source integration in public IT infrastructures, either by implementing several migration projects or by having processed clear, FOSS-specific policies, frameworks and support centers. Moreover, on several occasions local or regional authorities seem to implement more advanced policies (e.g. the case of Extremadura, the City of Freiburg migration project) than the ones defined by national frameworks or official guidelines and documents of the European Commission.

Local and regional authorities are often better positioned to directly integrate open source systems and applications in their internal processes and IT architectures by clearly defining needs and specifications through public tenders. By adapting open source solutions to regional contexts through extensive customization and localization they can also see immediate effects and improvements in administrative tasks or in services delivered to local communities.

6. Future work

This study serves to improve the knowledge of FOSS use in PAs. However, further room for research into FOSS, an increasingly important aspect of ICT adoption and growth in PAs is suggested. Further research on FOSS-adoption in PAs could include the quantitative study of FOSS-adoption and the study into the availability and perception of FOSS-vendors.

The future of FOSS-applications is that they are going to become much more prevalent, powerful and easier to use. Another important aspect that should be examined is whether FOSS must be judged on the same terms as proprietary software. The main issue is located on the fact whether calls for tender and other purchasing open source must be assessed on the basis of a realistic costing that takes account of all economic factors.

One more proposal for future work would be the provision of policy recommendations on issues and challenges pertaining to the use of FOSS by European PAs. The aim of this policy recommendation work would be the contribution in providing policy orientations and proposed actions that can help governments, PAs and European institutions fully harvest the benefits of FOSS.

References

CENATIC, Criteria for adopting open source software in public administrations, (http://observatorio.cenatic.es/index.php?option=com_content&view=article&id=723:criterios-para-adoptar-el-sfa-en-la-administracion-publica&catid=5:administraciones-publicas&Itemid=21).

CIO-Zone, CIO-SUMMIT, 5 Factors for Open Source Success, Published by govtech.com.

Danish Board of Technology Open-Source Software – in e-government, analysis and recommendations drawn up by a working group under the Danish Board of Technology, October 2002.

Dedrick, J., West, J., 2007. Movement ideology vs. user pragmatism in the organisational adoption of Open Source Software. In: Kraemer, K., Elliott, M. (Eds.), Computerization Movements and Technology Diffusion, From Mainframes to Ubiquitous Computing. Medford, Information Today.

Floss Final Report Part 1, Free/Libre Open Source Software: Survey and Study Evidence from Germany, Sweden and UK, Use of Open Source Software in Firms and Public Institutions, Berlin, July 2002.

Hars, A., Ou, S., 2000. Working for free? Motivations of participating in F/OSS projects, in: Proceedings of the 34th Hawaii International Conference on System Sciences, IEEE.

Holck J., Holm Larsen M., Kuhn Pedersen M., 2005. Managerial and Technical Barriers to the Adoption of Open Source Software, COTS-Based Software Systems, Springer.

Hwang, S., 2005. Adopting open source and open standards in the public Sector: five deciding factors behind the movement. Michigan Journal of Public Affairs, 2, Summer 2005.

James, S., Van Belle, J., 2008. Ensuring the long-term success of OSS migration: a South African exploratory study, 6th Conference on Information Science Technology and Management.

Kwan, S., West, J., 2005. A Conceptual Model for Enterprise Adoption of Open Source Software. Open Season, Sheridan Books, The Standards Edge, pp. 51–62.

Moolman, L., 2011. A characterisation of Open Source Software adoption decisions in South African organisations, March.

Morgan, L., Finnegan, P., 2007. How perceptions of Open Source Software influence adoption. Proceedings of the 15th European Conference on, Information Systems.

Olsoon, C., Ohrwall, Ronback, A., 2010. Collaborative development of public information systems: a case study of "Sambruk" e-services development. In: Paul Cunningham, Miriam Cunningham (Eds.), eChallenges e-2010, Conference Proceedings, IIMC International Information Management Corporation.

Oman, P., Hagemeister, J., 1992. Metrics for assessing a software system's maintainability. Software Maintenance, Proceedings Conference on, Nov 1992, pp. 337–344.

Paudel, B., Harlaka, Shrestha, J., 2010. Open Technologies and Developing Economies, CAN InfoTech.

Public Sector Forum, Open or Closed? A Survey of Open Source Software in Local Government, August 2009.
Rentocchini F., Tartary D., 2007 Open Source Software in the public sector: results from the Emilia-Romagna Open Source Survey (EROSS).
Richter, D., Zo, H., Maruschke, M., 2009. A comparative analysis of Open Source Software usage in Germany, Brazil and India. Fourth International Conference on Computer Sciences and Convergence Information Technology. 2009.
Salmivalli, L., Nissila, J., 2004. Curing health case information systems with Open Source Software. ECIS 2004 Proceedings, Paper 108.
Varghese S., 2003. Statskontoret – the Government's survey support, Swedish Agency for, Public Management.
Ven, K., Verelst, J., 2009. The importance of external support in the adoption of open source server software. In: Editor Open Source Ecosystems: Diverse Communities Interacting. Boston, Springer, pp. 116–128.
Ven, K., Verelst, J., Mannaert, H., 2008. Should you adopt open source software? IEEE – Software 25 (3), 54–59.
Wheeler, D.A., How to Evaluate Open Source Software/Free Software (OSS/FS) Programs, August 2011.
Wong K, Sayo P., 2004. Free/Open Source Software. A general Introduction, IOSN International Open Source, Network.