European Union
European Regional Development Fund

OSEPA
Open Source software usage by European Public Administration

INTERREG IVC
INNOVATION & ENVIRONMENT
REGIONS OF EUROPE SHARING SOLUTIONS

Project acronym:	OSEPA
Project name:	Open Source software usage by European Public Administrations
Project code:	INTERREG IVC, 0918R2

Document Information:

Document title:	Technical efficiency guidelines for selecting between and among FOSS and proprietary SW solutions
Date of Delivery:	31.08.2011
Component:	CP3
Component Title:	Exchange of experiences dedicated to the identification and analysis of good practices
Component Leader:	The University of Sheffield (USFD)
Task:	T3.2 (subtask 3.2.2)
Task Leader:	Computer Technology Institute & Press "Diophantus" (CTI)
Distribution (Restricted/Public):	
Nature:	Report

History Chart

Date	Changes	Cause of change	Implemented by
	Initial Document	N/A	CTI

Authorisation

No.	Action	Partner	Date
1	Prepared	CTI	31/8/2011
2	Approved		
3	Released		

Disclaimer

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

All rights reserved

Executive Summary

This report constitutes a review of literature and studies on pre-existing comparative studies regarding the technical, social and organizational factors on Free and Open Source Software (FOSS) usage. It comes as an output of the OSEPA (Open Source Software Usage by European Public Administrations) project aiming to assess and promote the uptake of FOSS in public administrations.

OSEPA project is conducted from January 2010 – December 2012 as a part of Interreg IVC program. The project is financed by the European Regional Development Fund (ERDF) and the EU National Co-financing thematic programme. OSEPA aims to explore this potential through interregional cooperation and to cultivate a debate among public administrations in this direction with a view to:

- Analyse, promote knowledge and foster awareness on the main benefits and disadvantages, cost evidence and effectiveness resulting from FOSS adoption and use.

- Explore, identify, build consensus on the framework conditions enabling FOSS to become a technically, financially, legally viable alternative offering of IT solutions.

- Explore, identify, build consensus on and promote European, national and regional policies and approaches that may facilitate the emergence of FOSS as a mature and viable business model.

- Promote awareness, exchange and disseminate knowledge, good practice and case studies regarding technical, financial and legal aspects of FOSS adoption by European public administrations in order to reduce uncertainty, inertia and resistance-to-change that limit experimentation and adoption of FOSS software.

- Discuss and promote the adoption of internal policies, mission statements, methodologies and action plans facilitating European public administrations to experiment, exploit and benefit from FOSS solutions.

- The project dissemination activities will promote awareness on FOSS and on OSEPA activities targeting relevant segments of the public opinion and Officers and elected representatives of local regional and national administrations.

CONTENTS

LIST OF FIGURES

1 Introduction

1.1 Scope of the document

This report constitutes a review of literature and studies on pre-existing comparative studies regarding the technical, social and organizational factors on Free and Open Source Software (FOSS) usage. Based on the above review, the document aims to outline the main technical, social and organizational factors and the main barriers that affect the adoption of FOSS usage.

This document is a foreseen output of the Open Source Software Usage by European Public Administrations (OSEPA) project aiming to assess and promote the uptake of FOSS in territorial public administrations. One of the key objectives of the OSEPA project is to provide technical efficiency guidelines for selecting between and among FOSS and proprietary software solutions. Within this scope, this report adds to knowledge resources that can help public stakeholders understand the technical/social/organizational environment and reach informed decisions when selecting the appropriate software. It can also be useful for FOSS developers, users and communities who are either directly or indirectly involved in the software market.

More specifically, this report targets three main recipient groups as prioritised below:

1. European Public Administrations: elected representatives and senior administrative staff, IT and procurement managers.
2. FOSS developers and entrepreneurs: individuals involved in developing and supporting, distributing and marketing FOSS systems and applications.
3. FOSS communities: work-groups and collaboration teams developing,maintaining and supporting FOSS projects.

The report is structured as follows: Chapter 1 outlines the scope and context of this report. Chapter 2 presents the review of literature and studies on pre-existing comparative studies regarding the technical, social and organizational factors on FOSS usage, while Chapter 3 analyses the main factors that affect the FOSS usage (including the main benefits and inhibitors/barriers). Finally, Chapter 4 identifies the main guidelines for selecting between FOSS and proprietary software.

1.2 Terms and definitions

1.2.1 ***Defining software***

Software (SW) can be shortly defined as the executable code that controls computer behaviour and operations. The term is used, however, to describe a wide range of programming languages, applications, procedures and all related documentation resources. Software also refers to a full cycle of processes from basic architecture to development, packaging and distributing as show in Figure 1 below:

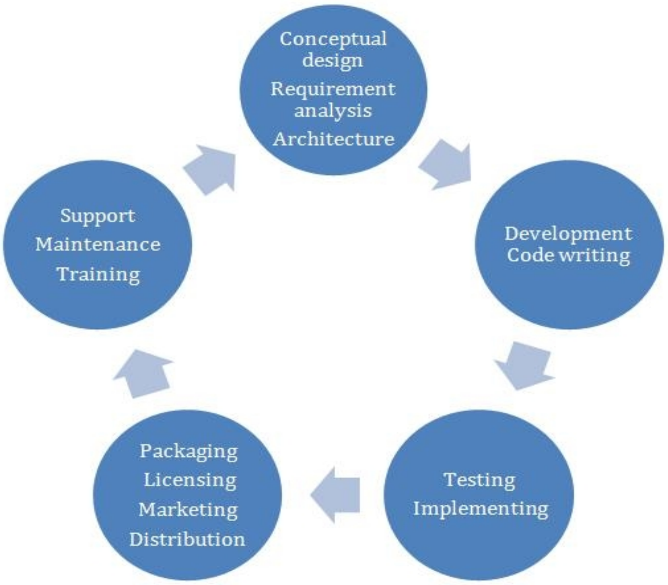


Figure 1: The software lifecycle

There are different types of software: operating systems and application software, middleware and embedded software. Throughout this document the term “software” refers to software used for operating and managing computer systems and applications, whether proprietary, free or open source.

1.2.2 Defining Free and Open Source Software

Although there are different definitions of free and open source software, there are some basic principles on which FOSS relies on. These refer to:

- the freedom to run a software program for any purpose
- the freedom to study and modify a software program by accessing its source code
- the freedom to distribute copies of a software program, whether modified or not

Additional prerequisites for open source software programs include: no discrimination against persons, groups or fields of endeavour and distributable, technology-neutral licences 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).

2 Review of Literature

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

2.1 Surveys on FOSS

2.1.1 *FLOSS Survey (Regions: Germany, Sweden and the UK) ([19])*

This survey was intended to yield information about FOSS use in several countries of the European Union. Due to budgetary restrictions, interviews could only be conducted for a limited number of countries (Germany, Sweden and the UK). While the first and the last represent significant markets in the European Union, the second is a typical case for a small country, which has in addition a high IT usage rate. Furthermore, especially Germany and Sweden were of interest, as desk research revealed that they show opposite extremes of FOSS usage: According to the last Internet Operating System Counter from April 1999 in Germany 42,7% of Internet hosts were running Linux, while the same figure for Sweden was only 16,9%.

To be able to compare the survey outcome by region, size or industry, the sample was stratified by eight strata or quota. Country, establishment size and industry were chosen as characteristics for determining to which stratum an establishment belonged. Indicator for size was the number of employees per establishment. Entities with less than 100 employees were not included in the sample.

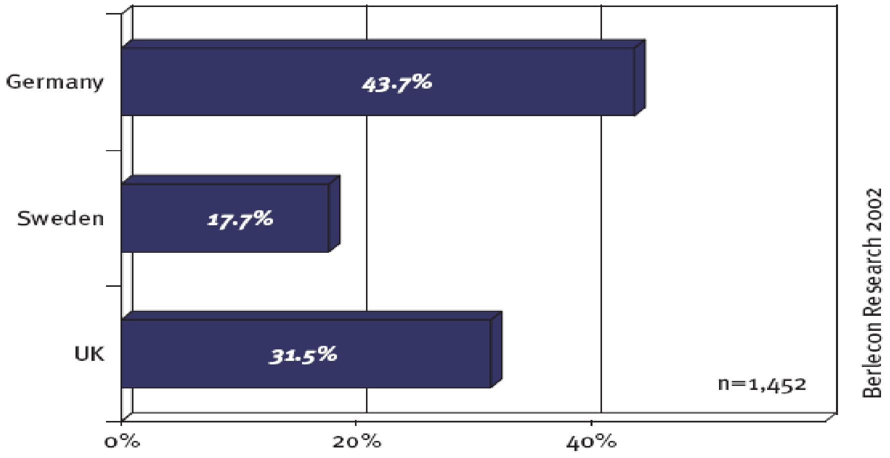


Figure 2: FOSS usage by country.

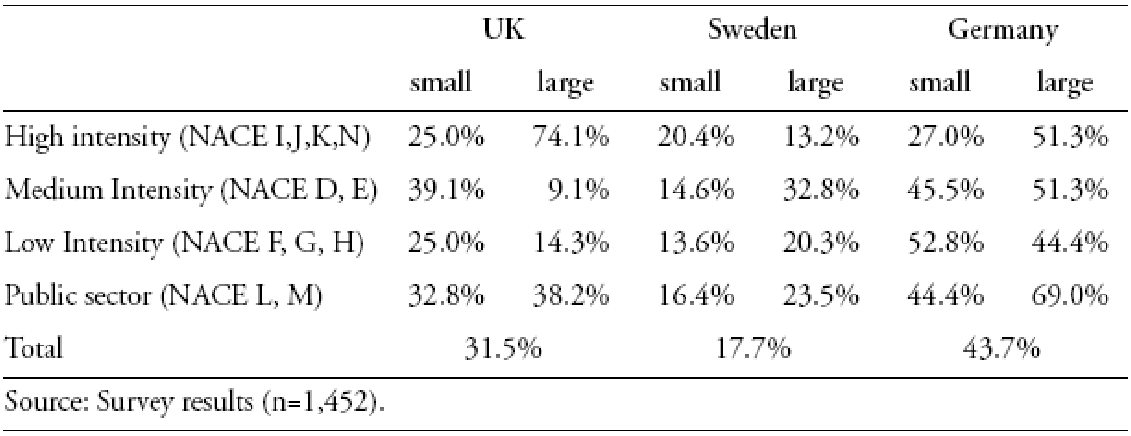
There are contained:

- establishments with 100 to 500 employees per unit.
- establishments with more than 500 employees per unit.

In addition, there were four sample quotas based on industries. There was a distinguish between the public sector and three quotas of the private sector. The private sector quotas were differentiated according to the amount of IT spending in relation to the revenues per industry. Motivation for this stratification was that industries with a high IT intensity - and thus high IT expenditures - might be more familiar with FOSS and might therefore show a different usage pattern from those with lower IT spending ratios.

In Figure 2 the researchers present the FOSS usage among Germany, Sweden and UK. For Germany and Sweden these numbers fairly accurately replicate those obtained by the Internet Operating System Counter (IOSC) in 1999. These project calculated Linux to be running on 42.7% of hosts in Germany and on 16.9% of hosts in Sweden. For the UK, the IOSC figured Linux to be running on 24.3% of hosts. Thus, the correlation is not as strong as for Sweden and Germany, although the ranking is the same.

Usage rates not only differ by country, but also within countries. For example, the FOSS usage rates of larger establishments are higher than those of small establishments in 8 of the 12 cells. This result is plausible since large establishments typically have a more diverse IT infrastructure increasing the probability that for some purpose FOSS is being used. One would therefore expect higher FOSS usage rates in these establishments.



	UK		Sweden		Germany	
	small	large	small	large	small	large
High intensity (NACE I,J,K,N)	25.0%	74.1%	20.4%	13.2%	27.0%	51.3%
Medium Intensity (NACE D, E)	39.1%	9.1%	14.6%	32.8%	45.5%	51.3%
Low Intensity (NACE F, G, H)	25.0%	14.3%	13.6%	20.3%	52.8%	44.4%
Public sector (NACE L, M)	32.8%	38.2%	16.4%	23.5%	44.4%	69.0%
Total	31.5%		17.7%		43.7%	

Source: Survey results (n=1,452).

Figure 3: Usage of FOSS in UK, Sweden, Germany.

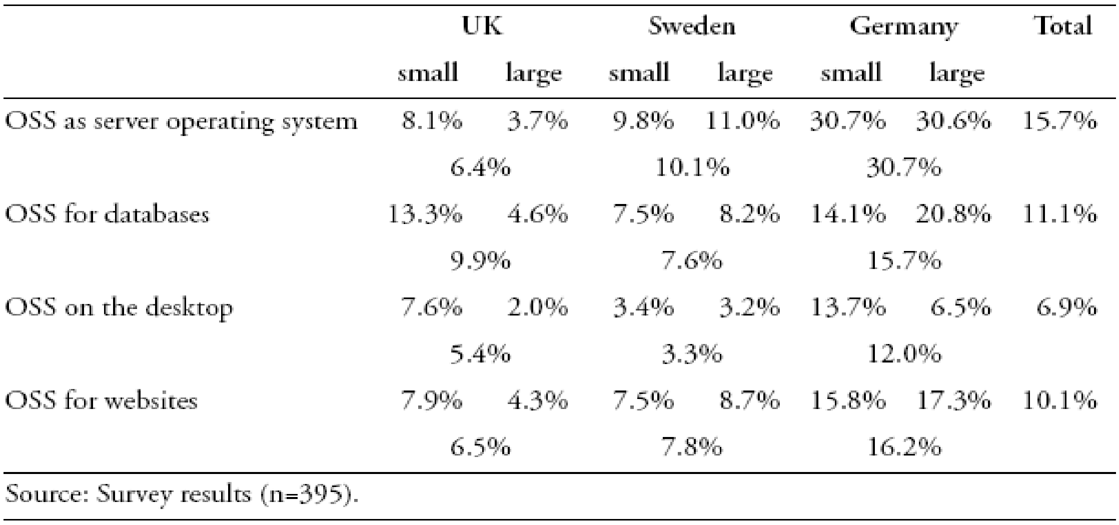
Quite consistently observable are above-average FOSS usage rates in the public sector. In 5 out of 6 cells is the FOSS usage rate higher in the public sector than on average in the respective country. There are also differences between the usage rates in the three different private sector segments. These are, however, not in any way systematic across countries and size classes.

The highest usage rate across all cells could be observed in large companies with high IT intensity in the UK (Figure 3). 74.1% of those companies contacted stated that they are using open Source software. The lowest rate - not counting the cell with only one FOSS using observation - could be observed within large companies with high IT intensity in Sweden (13.2%). This shows that one has to be very careful with generalisations about what sort of establishment tends to use FOSS.

One also has to be careful in interpreting the results as they rest on the assumption that those establishments that refused to participate and those that could not be reached differ not significantly in their FOSS use from those surveyed. Also it has to be considered that for the large UK companies with medium technology use only a single FOSS using observation exists.

Usage by IT area

Most popular is the use of FOSS as server operating system: On average 15.7% of establishments either currently use Open Source software like Linux or Free/Open BSD for server operating systems in regular IT operations or are planning to do so within the next year. As one can see, the differences between countries are considerable. While 30.7% of German establishments employ Open Source Software (OSS) this way, only 10.1% of Swedish and 6.4% of British establishments do.



	UK		Sweden		Germany		Total
	small	large	small	large	small	large	
OSS as server operating system	8.1%	3.7%	9.8%	11.0%	30.7%	30.6%	15.7%
	6.4%		10.1%		30.7%		
OSS for databases	13.3%	4.6%	7.5%	8.2%	14.1%	20.8%	11.1%
	9.9%		7.6%		15.7%		
OSS on the desktop	7.6%	2.0%	3.4%	3.2%	13.7%	6.5%	6.9%
	5.4%		3.3%		12.0%		
OSS for websites	7.9%	4.3%	7.5%	8.7%	15.8%	17.3%	10.1%
	6.5%		7.8%		16.2%		

Source: Survey results (n=395).

Figure 4: Usage by IT area.

Next in popularity is the use of FOSS for databases. MySQL, PostgreSQL, Interbase or SAP-DB are examples of such Open Source software. On average 11.1% of the establishments employ FOSS for databases. In this area, the differences are less pronounced. The OSS usage rate in Germany (15.7%) is slightly more than twice as high as in Sweden, where it is lowest with 7.6% (Figure 4).

On average 10.1% of the establishments use OSS in connection with creating or operating websites. There is a large variety of applications that are used in this area, e.g. Apache, PHP, Perl, Python, Squid or Open Source content management systems. Again the usage rate is highest in Germany (16,2%) and lowest in the UK (6.5%).

Finally, Open Source software can also be used on desktop computers. Examples are Linux as a desktop computer operating system, desktop extensions like KDE or Gnome but also application programs like Mozilla or StarOffice / OpenOffice. However, FOSS is not used very frequently on desktops. On average only 6.9% of the establishments in the three countries investigated use of FOSS on desktops – and this does not mean that they use FOSS on all their desktops. Again, the usage is highest in Germany (12%) and very low in Sweden, where only 3.3% of establishments use OSS on some of their desktop computers.

In all usage areas except the use of FOSS on desktops there is no clear indication across countries of a higher Open Source software usage rate by smaller or by larger establishments. Only the use of FOSS on desktops is more frequent in smaller than in larger establishments (in Sweden not significantly).

General attitudes to OSS vs. specific buying decisions

An enterprise’ s or organisation' s decision to use FOSS can be driven by two sorts of motives. The first sort of motives is application specific, e.g. an expected greater stability or lower costs for that specific application in comparison to its commercial alternatives. The second sort of motives is more general, like the wish to support the Open Source community by using Open Source software or by letting one’s IT personnel work on OS projects on company time.

Figure 5 shows the weighted mean of answers to the different statements. As the figure shows, agreement is strongest with the statement that establishments use OSS to become more independent from the pricing and licensing policies of large software companies. The average answer is between “agree somewhat” and “neither nor”.

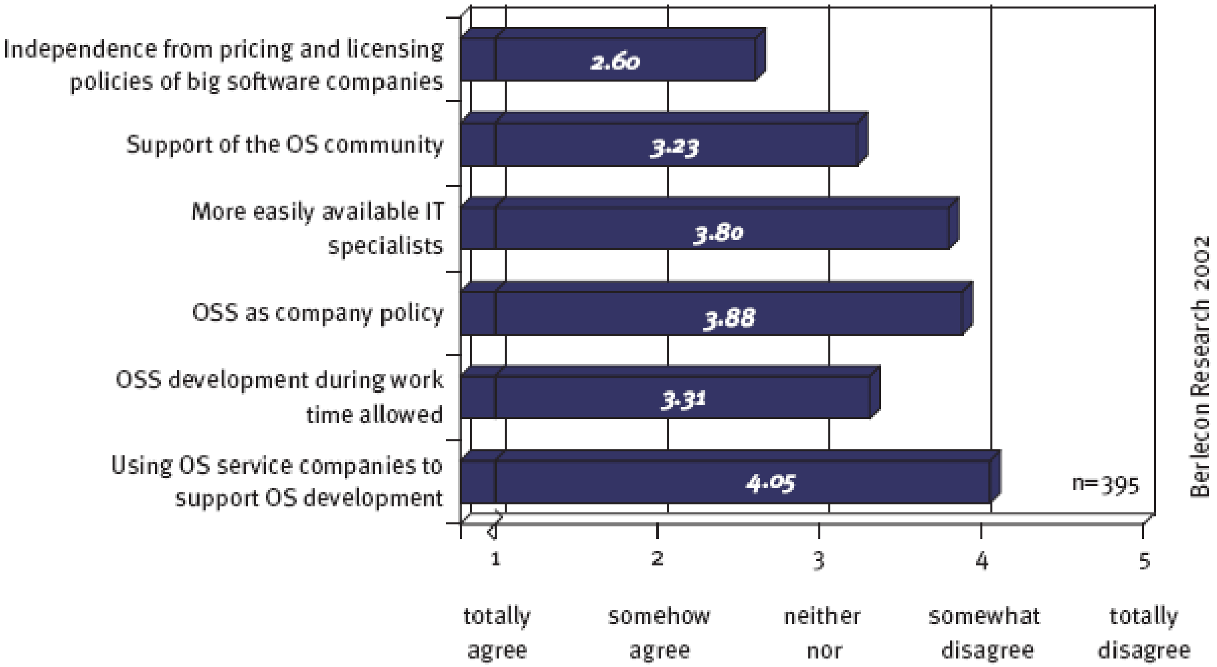


Figure 5: Attitude to Open Source software in general.

Next in order are different ways to support the OS community, either indirectly by using OSS or directly by letting one’s developers work on OSS development on company time. Nevertheless, these assessments are already more on the negative side, which shows that individual gains for the establishments are a much more important reason for using OSS than the altruistic wish to further OSS development or to support the OS community.

The least agreement was on average found with the statement that companies might use OSS because IT specialists are more easily available, with the statement that OSS use might be company policy as well as with the statement that establishments might work together with OSS service companies to support OSS development.

2.1.2 *Italy: Emilia-Romagna Open Source Survey (EROSS 2006) ([47])*

This survey was conducted in the region of Emilia – Romagna, Italy. The first striking result, shown in Figure 6, points to the presence of unaware FLOSS adopters, i.e. municipalities answering that they do not have FOSS (in this survey the similar term Free / Liberty Open Source Software (FLOSS) is used without losing in accuracy) installations and, at the same time, choosing some open source applications. This pattern is presented in results from FLOSSPOLS as well as in [23].

This result is a hint on the small amount of knowledge available in the area of FLOSS to Emilia - Romagna municipalities. In addition, this fact points out the important role of information in the area of FLOSS, meaning that policies aiming at increasing the level of knowledge about FLOSS are the most welfare increasing ones The percentage of FLOSS adopters inferred by EROSS 2006 survey comes out to be quite high, i.e. 70% of respondents are found to adopt FLOSS.

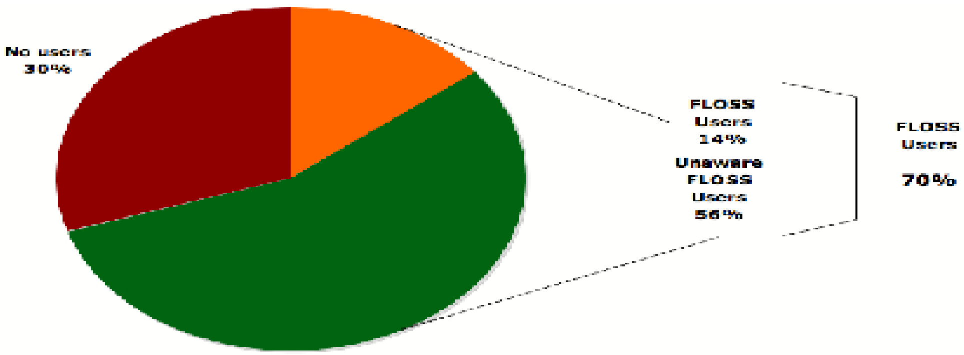


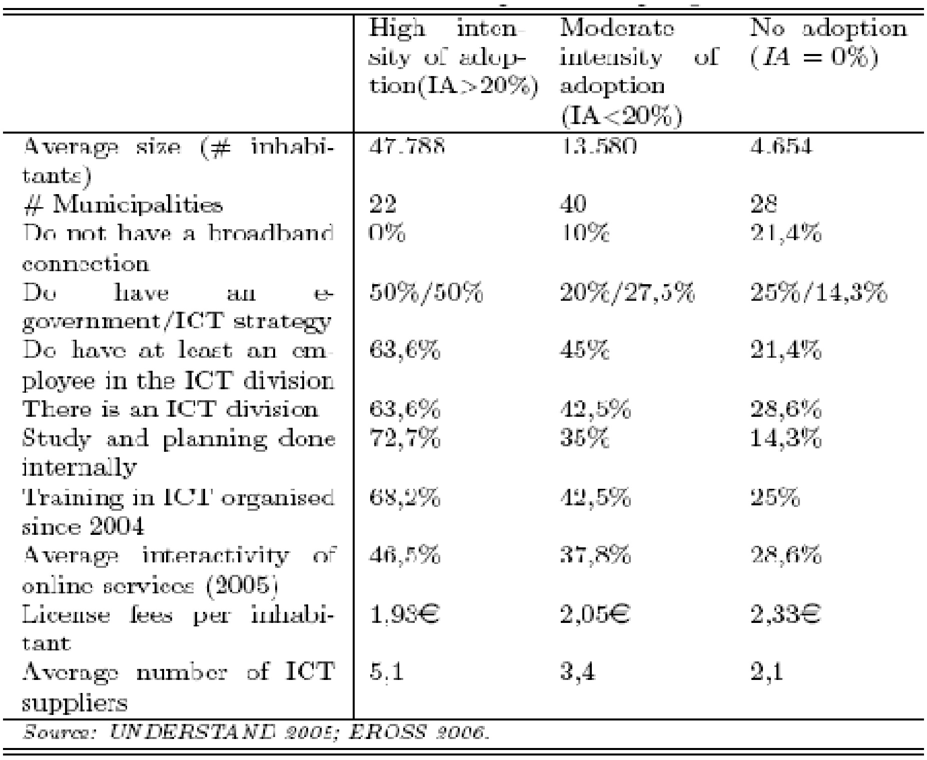
Figure 6: Municipalities adopting FLOSS

The survey found a percentage of FLOSS adopters lower than 38%. On the contrary, FLOSSPOLS reports similar statistics concerning the percentage of FLOSS adopters, namely almost 80% over a total of 955 European local governments. The higher value of the survey estimates, as well as the one coming out from FLOSSPOLS study, must be attributed to a self-selection mechanism by which users most interested in FLOSS are more likely to have answered to the questionnaire.

Indeed, while in specific studies dealing with FLOSS, the number of FLOSS adopters has been found to be quite high, in other studies not addressing specifically this topic, the percentage of users falls considerably.EROSS questionnaire has been planned to be integrated with data collected through another survey. Merging the two datasets allowed to display a clear picture of the characteristics of municipalities according to their FLOSS intensity of adoption.

In Figure 7 the survey first divides all the municipalities by the total intensity of FLOSS adoption (henceforth IA), i.e. no adoption (IA= 0%) moderate adoption (IA<20%) and high adoption (IA>20%). In this way, the survey able to underline differences in the characteristics of the municipalities both in their class of adoption and among them. The survey can derive the main characteristics of the municipalities adopting intensively FLOSS from column 1, in Figure 7. Indeed, municipalities with a high intensity of FLOSS adoption has, on average, a larger size, they are furnished with a broadband connection and they have adopted an e-government/ICT strategy.

Furthermore, the presence of a formal ICT structure, the ability to develop software internally and, finally, ICT training for employees are all relevant features. So, it looks like that the intense adoption of FLOSS discriminates between those municipalities which see ICT as an important strategic support for institutional activities and those which are not able to, or do not want to, go into this direction.

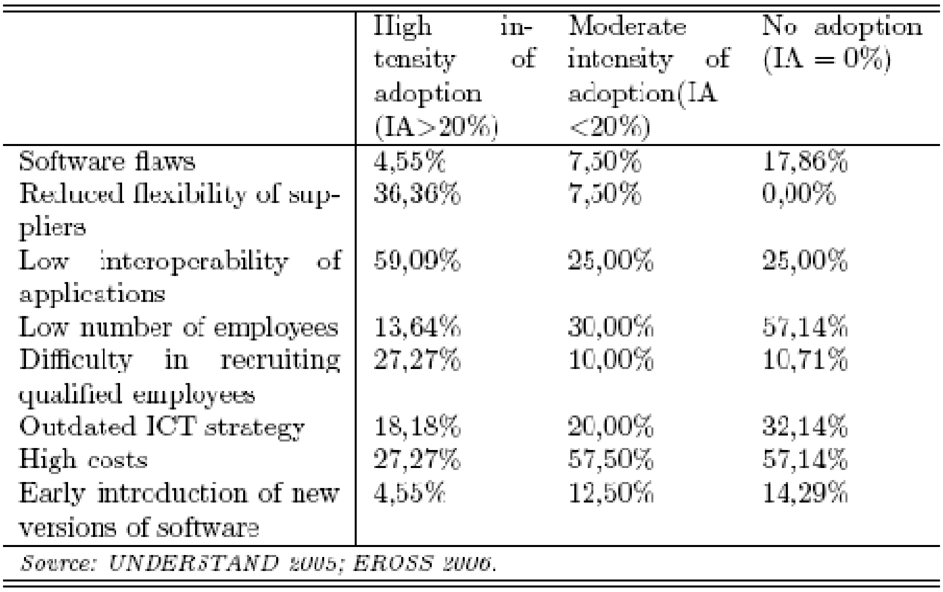


	High intensity of adoption(IA>20%)	Moderate intensity of adoption (IA<20%)	No adoption (IA = 0%)
Average size (# inhabitants)	47.788	13.580	4.654
# Municipalities	22	40	28
Do not have a broadband connection	0%	10%	21,4%
Do have an e-government/ICT strategy	50%/50%	20%/27,5%	25%/14,3%
Do have at least an employee in the ICT division	63,6%	45%	21,4%
There is an ICT division	63,6%	42,5%	28,6%
Study and planning done internally	72,7%	35%	14,3%
Training in ICT organised since 2004	68,2%	42,5%	25%
Average interactivity of online services (2005)	46,5%	37,8%	28,6%
License fees per inhabitant	1,93€	2,05€	2,33€
Average number of ICT suppliers	5,1	3,4	2,1

Source: UNDERSTAND 2005; EROSS 2006.

Figure 7: Municipalities adopting FLOSS

Figure 8 presents the elements that have been selected as the main obstacles for a correct adoption of the ICTs in the PAs, relying on the same classification used in Figure 7.



	High intensity of adoption (IA>20%)	Moderate intensity of adoption(IA <20%)	No adoption (IA = 0%)
Software flaws	4,55%	7,50%	17,86%
Reduced flexibility of suppliers	36,36%	7,50%	0,00%
Low interoperability of applications	59,09%	25,00%	25,00%
Low number of employees	13,64%	30,00%	57,14%
Difficulty in recruiting qualified employees	27,27%	10,00%	10,71%
Outdated ICT strategy	18,18%	20,00%	32,14%
High costs	27,27%	57,50%	57,14%
Early introduction of new versions of software	4,55%	12,50%	14,29%

Source: UNDERSTAND 2005; EROSS 2006.

Figure 8: Obstacles to the adoption of ICT inside the PAs

In Figure 8 the survey presents some obstacles to the adoption of ICT inside the Public Administrations (PAs). The survey noted immediately that there are differences among the three groups. Municipalities with a high intensity of FLOSS adoption (column 1) 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 (column 2) and no intensity (column 3), main obstacles are the low number of employees and high costs. The figure shades light on both advantages and disadvantages on the adoption of FLOSS in Emilia - Romagna municipalities.

Indeed, PAs with a high intensity of FLOSS adoption deem both software flaws and high costs as less stringent obstacles for the implementation of a proper ICT strategy. This is an empirical proof of the theoretical arguments in favour of FLOSS, which see technical superiority together with savings from license fees as the main motivations to foster its adoption. On the other side, the main drawbacks concerning FLOSS adoption are found to be the low flexibility of suppliers together with the interoperability of applicants. These differences in perceived obstacles can be interpreted as the causes that pushed some of the municipalities interviewed to experiment and, in a second instance, to adopt.

Figure 9 shows the intensity of FLOSS adoption in the area of client. Desktop systems, e-mail clients and web browsing are the four sub - classifications that have been individuated in the area of client/desktop.

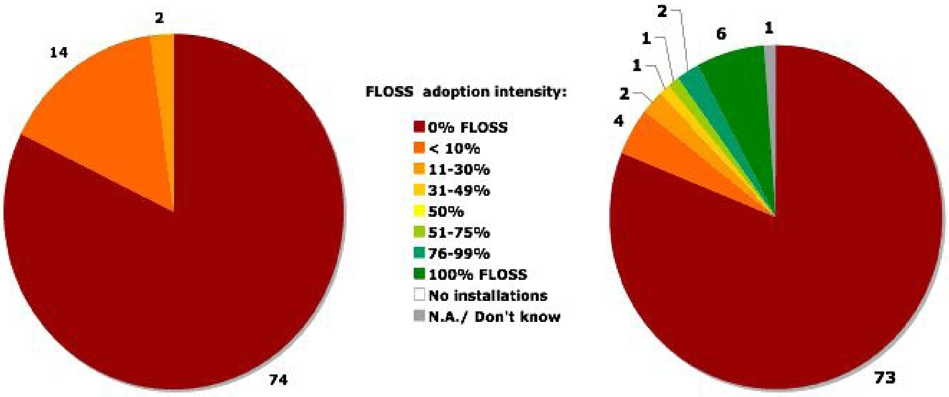


Figure 9: Software client/desktop (# Municipalities)

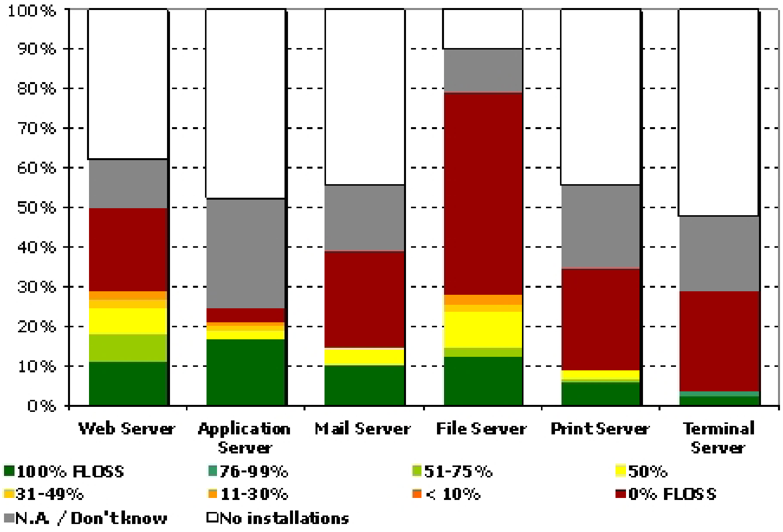
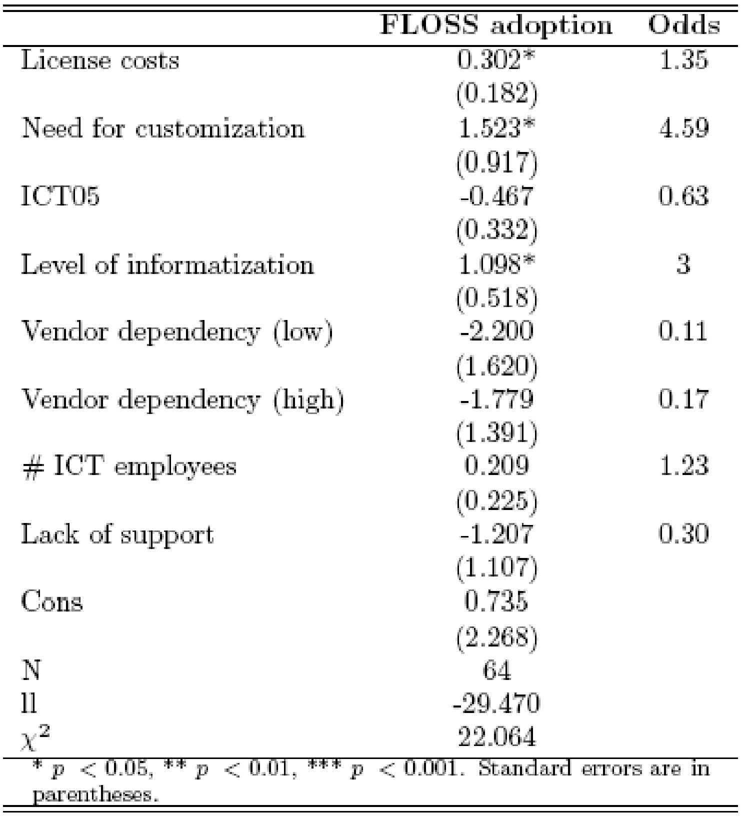


Figure 10: Categories of FOSS in Municipalities.

FOSS is very complex and this implies that the process could be slowed down by several obstacles. This is why this figure should be interpreted as very promising. When the attention is shifted to web servers, then interesting figures are found. In Figure 10 the intensity of FLOSS adoption on the server side relative to the percentage of municipalities is displayed. It is worth noting that few municipalities have internalized web server management and server in general. Nevertheless, among them the majority adopts FLOSS solutions.

On the basis of the data gathered through the survey an econometric analysis has been conducted that contributed to the understanding of the effect of FOSS adoption on interactive public services. In particular, the main concern has been the estimation of the impact of FLOSS adoption, together with a set of other relevant variables, on the level of interactivity of a single Public Administration. After that, the factors that are likely to explain the decision to adopt FLOSS by the municipalities of Emilia - Romagna were taken into consideration. Main results from the estimation are presented in Figure 11.

The main research question deals with the impact that FOSS is likely to have on the level of interactivity of public services. The main idea behind this standardized procedure deals with the measurement of the level of interactivity characterising twenty basic public services.



	FLOSS adoption	Odds
License costs	0.302*	1.35
	(0.182)	
Need for customization	1.523*	4.59
	(0.917)	
ICT05	-0.467	0.63
	(0.332)	
Level of informatization	1.098*	3
	(0.518)	
Vendor dependency (low)	-2.200	0.11
	(1.620)	
Vendor dependency (high)	-1.779	0.17
	(1.391)	
# ICT employees	0.209	1.23
	(0.225)	
Lack of support	-1.207	0.30
	(1.107)	
Cons	0.735	
	(2.268)	
N	64	
ll	-29.470	
χ^2	22.064	

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$. Standard errors are in parentheses.

Figure 11: The determinants of FLOSS adoption

The second step of the analysis consists in determining which factors affect PA's decision to adopt FOSS solutions. This part must be interpreted as an empirical test of the findings from FLOSSPOLS study [23]. Results derived from the latter indicate a set of factors as essential in explaining the decision of European PAs to adopt or not to adopt FLOSS solutions.

In particular, seven main factors are put forward:

1. high costs of license fees;
2. need for customisation
3. dependency from the suppliers of proprietary software
4. high training costs needed for new FLOSS adopters
5. lack of support for FLOSS solutions
6. server management advantages thanks to high technical performance of FLOSS compared to proprietary solutions
7. the level of interoperability among different applications.

Analytical results are displayed in Figure 11.

Three variables result to be significant and to explain positively the adoption of FOSS solutions by PAs in Emilia - Romagna. The variable with the stronger effect is the one proxying for the need for customisation. Indeed, when the PA perceives an impelling necessity to customize its software products in order to tailor them to user specific needs, then the odds of adopting FLOSS increase by a factor of 4.59.

The level of informatization contributes to increase the rate of FOSS adoption as well. An increase of 1 point in the level of informatization of a PA brings the odds to increase by a factor of 3. This is fairly reasonable given the capacity of FLOSS solutions to improve the management of more sophisticated information systems. Finally, more expensive license fees for proprietary solutions leads the odds to increase by 1.35.

This is a well known advantage that FLOSS provides compared to proprietary solutions, that is no license fees have to be paid. Hence, the survey found strong empirical evidence for patterns that have been only mentioned by the literature so far.

2.1.3 *Sweeden: Swedish Association of Municipalities for Joint Development of Public e-Services ([43])*

Sambruk (Swedish Association of Municipalities for Joint Development of Public e-Services) has conducted a survey about open-source software in public administration.

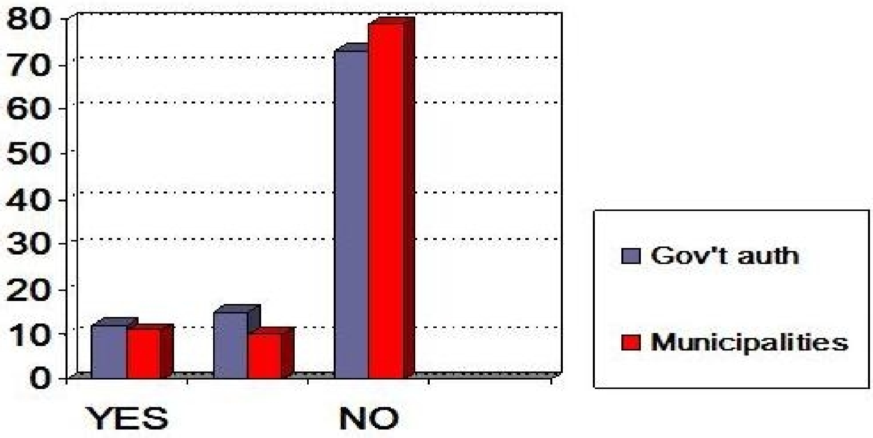


Figure 12: Existing FOSS Procurement policy

The results are presented below.

(In red colour the results that concern the municipalities, are presented.)

- **How common is the usage of free / open source software (FOSS)?**

  Positive answer was given by 60% of the government authorities and by 40% of the municipalities. Analytically :

  50% large authorities

  13% mid-sized

  3% small

- **Which are the areas of FOSS utilisation?**

  Infrastructure

  69% 70% operating systems

  53% 32% databases

  51% 26% internet applications

Applications
49% 18% application platforms
34% 10% business applications
23% 34% office applications
21% 8% administrative applications

- **Which are the obstacles of FOSS usage?**

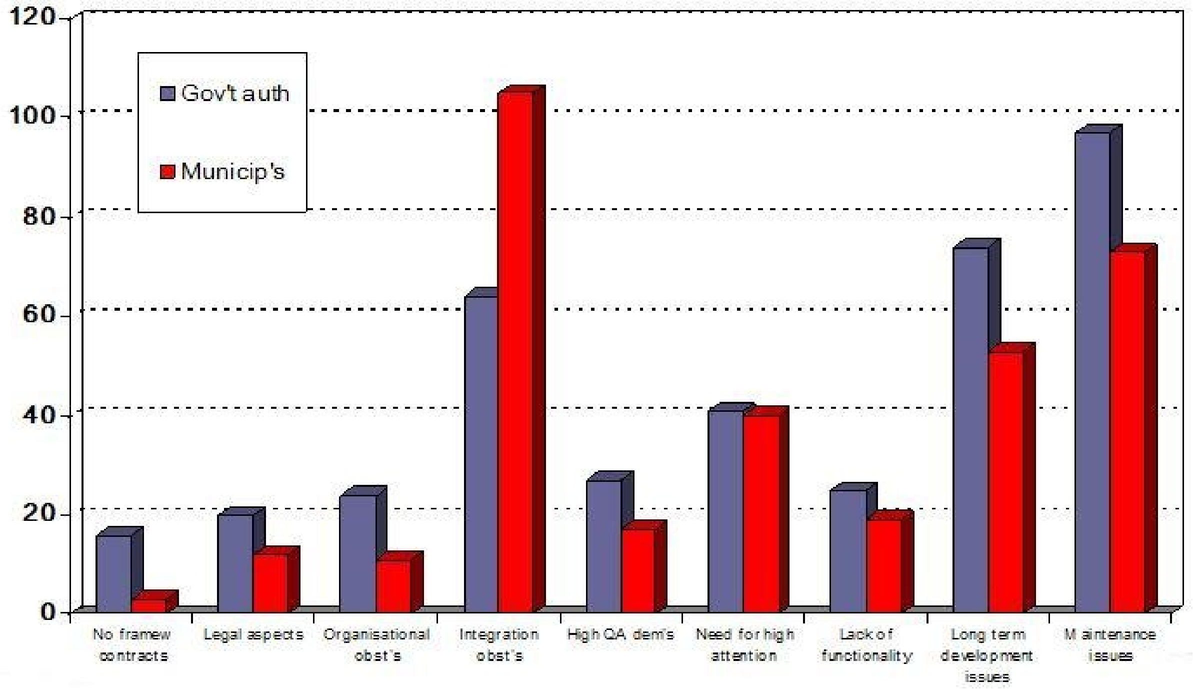


Figure 13: Obstacles of FOSS usage.

- **Is there the need for support in procurement & utilisation of FOSS?**

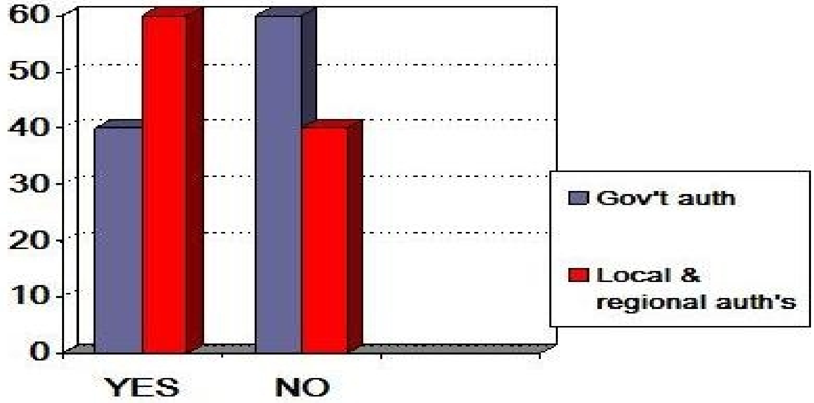


Figure 14: Need for support in procurement & utilization of FOSS.

Government authorities
50% - large
40% - mid-sized
30% - small

- **What is the type of the needed support?**

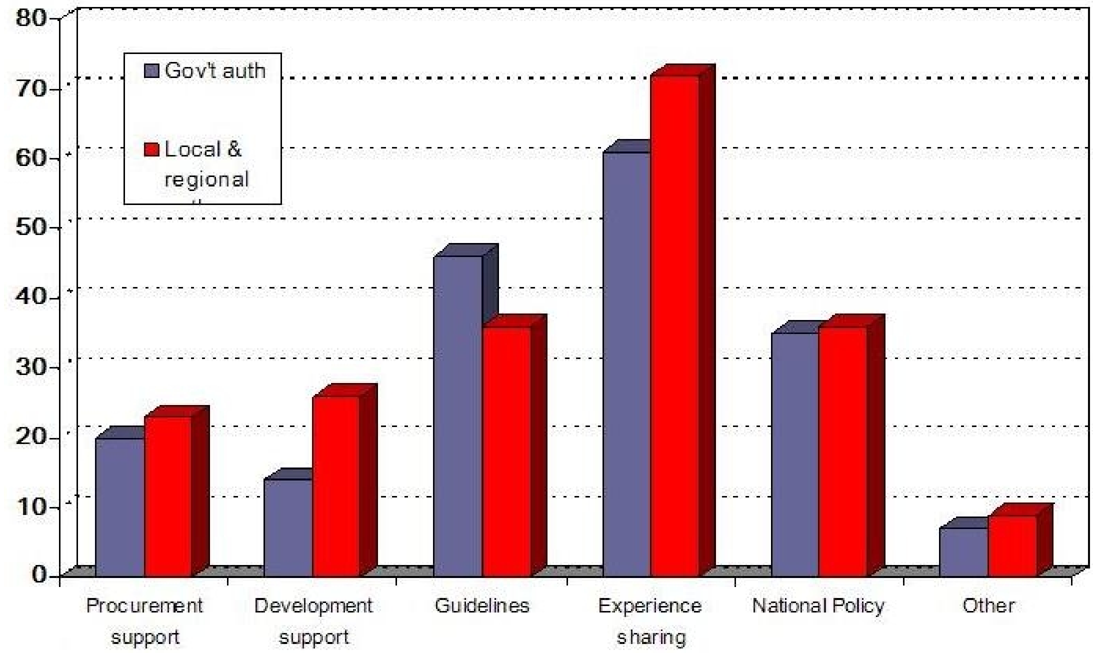


Figure 15: Type of needed support.

Type of needed support

- Procurement support
- Development support
- Guidelines
- Experience sharing
- National Policy
- Other

2.1.4 *Open Source Software in the Public Sector: Policy within the European Union (FLOSS - Deliverable D18: FINAL REPORT - Part 2B) ([20])*

This survey investigates the policies that are followed in the public sector of certain European regions.

France

Developers with a French citizenship were with 16.3% the highest amount of respondents in the FLOSS developers' survey. 91.5% of them stay in France, the rest is living in other countries of the European Union (4.8%), USA (1.7%), or the rest of the world (2.0%). All together France has with -1% a negative migration balance. 15.1% of all French open source programmer declared to have regular contact with more than ten other developers in the community. Thereby French developers are in that category slightly less in contact than the worldwide average (17.5%) and much less in contact than US American open source programmers (22.1%).

In the next category, open source developers being in contact with three to ten other developers, France is with its 43.0% over the worldwide average (38.9%) and much over the compared US group (32.1%). 26.6% of all French developers are with one of two other open source developers in regular contact. This group is exactly comparable to the worldwide average (26.3%) and slightly more than the US American developers group (25.1%). 15.3% of all French developers have no regular contacts to the open source developers scene at all. This group is 2% under the worldwide average (17.3%) and much smaller than the compared US group (20.7%). 7.3% of all developers in France have lead four or more open source software projects and are thereby over the worldwide average (7.0%).

The situation is quite comparable to the USA (7.3%). 54.6% of the French developers have less leadership experience, up to three projects, whereas on a worldwide average 58.1% and in the USA 55.5% of all developers have lead up to three projects. 38.1% did not lead a project at all (worldwide 34.9%, USA 37.2%). France strong engagement of the open source software developer community as well as the strong governmental policy towards open source software will lead to more implementation of open source software in the public sector. The political pressure towards open standards could lead - even if legally not enforced - to their realization in the public sector. This role of the state as a grantor of software interoperability would most probably lead to a strong growth of the open source movement.

Germany

Developers with a German passport rank with 12.4% second in the list of nationalities. 92.6% of them are currently living in Germany, the rest is disseminated throughout the European Union (4.0%), the USA (1.1%) and other countries (2.3%). With 0.2% Germany has a slight positive migration balance. Only 12.9% of all German open source developers are in regular contact with more than ten developers in the scene. 41.0% have contact with three to ten developers. The next group, one or two contacts, is the greatest in our sample (29.1%). 17.0% of the German open source developers have no regular contacts with other members of the community.

Thereby Germany is one of the least directly connected countries within open source programmers worldwide. 7.1% of all German developers are highly involved in leadership (four and more projects). 60.2% have lead up to three projects, whereas

32.7 of all German open source software developers have participated in projects only in a non-leading function. In terms of implementation of open source software into public sector institutions, Germany takes up - together with France - the leading role inside and outside Europe.

The German Parliament in general demands the usage of open source software in federal administration. In 2001 it decided that open source products should be used wherever costs could be decreased. After France, Germany has the second largest community of open source software developers responding to the FLOSS Developer Survey (other surveys show Germany as the country with the highest number of open source software developers). Governmental organizations show strong interests to support open source software in the public sector. Driving factor is, in the main, savings in expenditure. Growing interoperability is, even if perceived as very positive, more a side effect in importance. Nevertheless the practically oriented policy and the strongly increasing implementation of open source software in the public institutions will contribute to an augmentation of open source software projects also outside the German public sector. This will have a strong impact on the usage of open source software also in the private sector.

All the results for the various regions are presented in Figure 16.

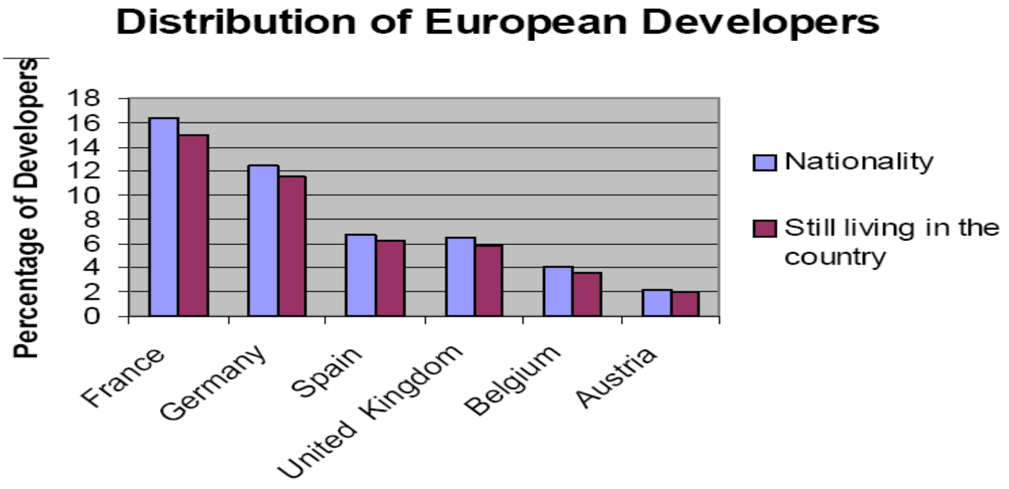


Figure 16: Distribution of European Developers

Spain

6.7% of survey respondents claimed Spanish nationality, making Spain among the top European countries for open source software development. 93.1% of all open source developers who declared their nationality as Spanish are living in Spain. 3.4% are living within another member state of the European Union, 2.1% live in the USA, and 1.4% live somewhere else. Spain's migration balance is 0.3% negative. Only 8.7% of the Spanish open source developers have no regular contact with other community members. This is the smallest ratio of the whole sample. 26.1% of the developers mentioned one or two regular contacts with scene. In the middle range (three to ten contacts).

Spain is with its 45.6% of developers far over the average and also with its 19.6% of developers who have contact with more than ten other open source participants Spain is ahead of the worldwide average. In general, Spanish developers can be regarded as among the most well-connected. 8.7% of all Spanish open source software developers are very experienced in leadership (four or more projects). 65.9% of the Spanish programmers declared that they lead up to three projects. And only 25.4% stated that they did not lead any open source project.

In a region of Spain, the border province to Portugal, Extremadura, the regional government adopts Linux as the official operating system within schools. The 670 schools are based on open source software. The training of the 15.000 Extremadura' s teachers on the system is now the main priority of the government within this project. Despite the low support of administrative policy change in the public sector seems not to be improbable. The developers' activity in the private sector seems to be Spain's trump card. Not only the high amount of community members, but especially their high experience in project leadership and the high degree of connectedness will help to boost Spain's open source movement.

United Kingdom

6.5% of all open source software developers are British citizens. In that sense it is quite comparable to Spain. 88.7% of the developers with British nationality are living in the UK, with the rest distributed across other European Union states (4.9%), the USA (3.6%) or other countries (2.8%). With 0.1% positive migration balance the United Kingdom is almost balanced regarding its immigration versus emigration. In the UK 21.8% of all open source developers have more than 10 regular contacts in the scene. This gives the UK the largest proportion of highly connected developers in our sample. In the middle range (three to ten contacts per developer) UK developers are far on the bottom end (33.2%) and within the category of up to two contacts the UK has 23.9%. 21.1% of the UK developers have no contact with other programmers on a regular basis. This is also the highest rate within our European sample. 7.1% of UK's open source software programmers are very experienced in leadership (four and more projects), 56.7% have led one to three projects, and 36.2% did not lead projects so far. Implementation of open source software is, in the main, concentrated to the national health care system.

Figure 17 shows a graph that presents open source developers in regular contact with other developers.

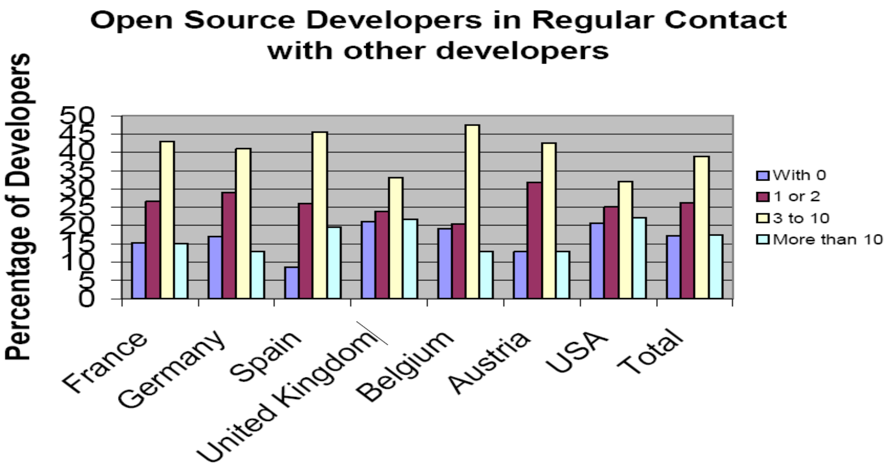


Figure 17: Open Source developers in regular contact with other developers

Belgium

4.0% of participants of the developers' survey stated to be Belgium citizens. Therefore the country ranks relatively low in terms of absolute figures. However regarding its small population Belgium's density of developers is considerable high. 88.1% of developers with Belgian nationality live in the country. The rest is residential in other member states of the European Union (10.7%) and the United States (1.2%). This quite high degree of mobility is in part certainly explainable by Belgium's central geographical situation within the European Union, by its small size, as well as by its multilingual population. Belgium's migration balance is with its -0.4% comparatively high, since this is measured on all open source software developers and would mean-10% compared to Belgian open source developers. The country's share of top connected open source developers (more than ten regular contacts) is comparatively low (12.8%), whereas the middle range (three to ten contacts) is with 47.5% over the average. 20.5% of all Belgian open source developers have regular contact with one or two other members of the community; 19.2% are not at all in regular contact with the community. 6.4% of all Belgian developers have strong leadership experience (four or more projects), 56.4% lead one to three projects, and 37.2% have no experience in leading projects at all.

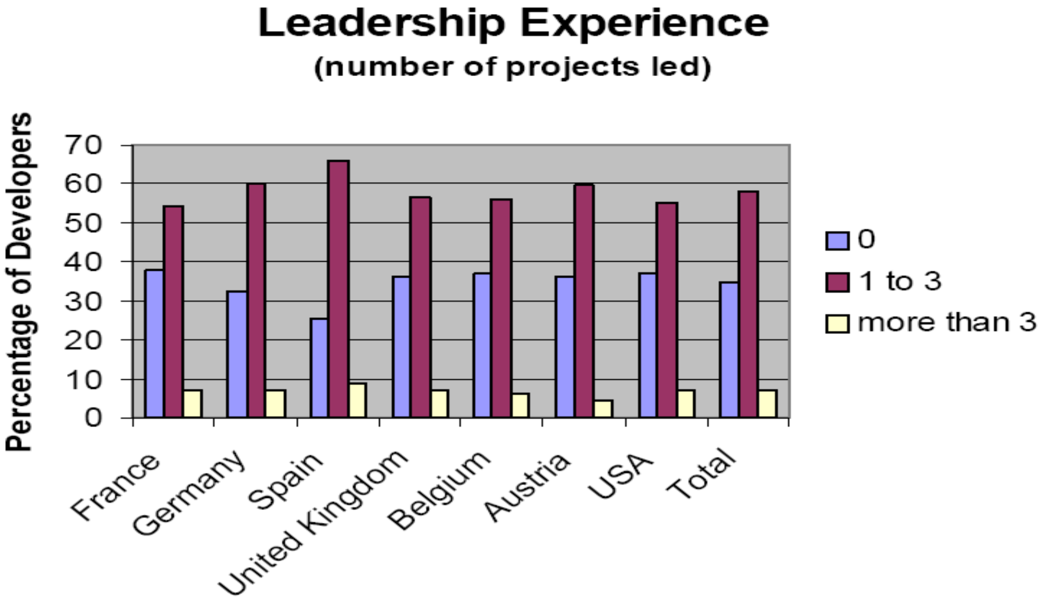


Figure 18: Leadership experience

The implementation of open source software in the public sector in Belgium is growing. Especially the growing inclusion of open source in public tenders looks promising for the future development of the open source software potential in Belgium.

The usage of open source software within the public sector of the Brussels region is expected to grow further and influence the public ICT strategy. The highly developed open source programmers' scene points in the direction of further growth of the movement, at least in the private sector of the country. The total amount of experience of each country is presented in Figure 18.

Austria

With 2.2% open source developers holding Austrian nationality, Austria is also in the lower category of the sample. Regarding its small population size, Austria's relative share in the open source community is however perceptibly high. Similar to Belgium, lot of these developers live abroad (10,7%). Most of them are residential in Germany (6.4%). The rest is distributed over rest of the world (4.3%). Different to Belgium, Austria has an equalled migration balance (0.0%). Regarding the connectivity, the country has comparatively not that many highly connected open source developers.

12.8% have more than 10 contacts on a regular basis. 42.5% are in contact with three to ten other open source programmer and 31.9% have one or two regular contacts in the scene. 12.8% of all open source developers in Austria have no regular contacts in the scene, a comparatively low value. Developers with strong leadership experience (four and more projects) are comparatively rare (4.3%). 59,6% of all Austrian developers have lead one to three projects, and 36.2% did not lead any open source project so far.

2.1.5 A Survey of Open Source Software in Local Government (UK) ([46])

Background

With the current economic downturn placing public finances under significant pressure, local authorities need to prepare for a more challenging future. In a climate of increasing budget constraints, councils are now facing inescapable demands to develop new and innovative ways to transform services, generate cashable efficiencies and deliver more for less. At the same time, local government IT costs are rising: Socitm (the professional association for public ICT management) reported ICT spending by UK local authorities would soar by 5% in 2008/09. These developments underline the need for councils to drive more value from their IT investments.

These imperatives are concurrent with a fresh resurgence of interest across UK government in open source software. Is local government, the sector seen as potentially most receptive to open source, ready, willing and able to embrace this change? What do councils see as the key strategic, management and technical barriers to engaging fully with open source? And how can these obstacles be best overcome?

To help find answers to these questions and others, Public Sector Forums, who host an online community of some 14,000 professionals who work in and around the public sector, conducted research to examine the adoption, perceptions and experiences of open source technologies within UK local authorities. The survey, which preceded the publication of the UK Government Action Plan for Open Source on 24 February 2009, ran from 18 November to 12 December 2008 and was completed by 168 respondents. Responses were anonymous unless respondents provided contact details. The majority of those taking part were from District Councils (31%), followed by Unitary authorities (27%), County Councils (17%), Mets (14%) and London Boroughs (7%). The survey looked at variety of issues including:

- Distribution of open source technologies in local government
- Attitudes within local authorities to open and proprietary software
- Predicted growth of open source in local authorities, including local government business/IT areas expected to be impacted greatest by open source

- Perceived and real risks, challenges and barriers to open source adoption in local government, as well as areas of opportunity
- What needs to be done to help councils address these obstacles and increase their use of open source software

Headline Findings

Though open source adoption in local government is wide and diverse, levels of usage have not yet extended to the same degree on the desktop. The survey shows Microsoft has an effective monopoly on the operating systems used on local government desktops. Aside from web and IT teams, the majority of local government staff are currently unlikely to interact knowingly with open source software in the workplace. Despite the widespread use of proprietary desktop software, there appears to be a distinct sense of concern within many local authorities about software licensing costs.

Open source office productivity software, however, has gained a small but significant foothold in a few local authorities. This is the single key area where councils believe open source will make the greatest impact over the next two years.

Two thirds of respondents (65%) believe their council needs to increase its use of open source software – with 27% of those surveyed strongly in favour. Most local government open source users are primarily attracted by the lower cost of open source software, followed by the potential freedom from supplier dependency.

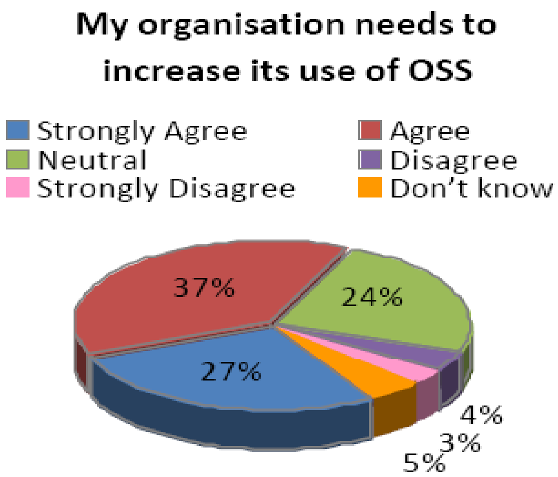


Figure 19: Organisation needs

Almost two-thirds of those surveyed believe the benefits of open source generally outweigh the 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 (Figure 19). The majority view (42%) is that local authorities will increase their use of open source software over the next three 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.

Open Source Adoption Trends in Local Government

The majority of those surveyed (43%) believe the next three years will see their local authority increase its use of open source software. Most foresee a moderate rather than significant rise in usage. At the time of the research, which preceded the Government's announcement on its Open Source policy, around a third (37%) of respondents expected their council's level of open source adoption will remain unchanged. Just 3% predicted open source would be used less. Almost a fifth of responses fell into the 'don't know' category, maybe highlighting the sense of uncertainty in some areas of local government around potential adoption of open source software and a 'wait and see' approach.

Some two-thirds of those surveyed (64%) believe their organisation needs to increase its adoption of open source software, with a quarter of all respondents (27%) strongly in favour of such an approach. However this does indicate a third who were either neutral (24%) or disagree (7%) with councils using more open source, suggesting there are a substantial contingent in local government who still need to be won over.

With a few notable exceptions, the research found the standard desktop PC suite provided by local authorities to their staff appears to be almost exclusively a Microsoft environment. Of the local government users surveyed, an overwhelming 98% stated the PC provided by their employer ran a Microsoft Windows operating system. Alternative operating systems, such as Apple and Linux distributions, are currently a rarity on mainstream office PCs in local authorities. An overwhelming 83% of respondents said their PC used the older Microsoft Windows XP operating system, while just 4% indicated their council had transitioned to Windows Vista.

How Open Source Systems are Used in Local Government

Open source software is already used widely by local authorities at many levels and for a very large number of different applications. The research shows, primarily, these are for server-based and publishing/content management purposes – for instance the LAMP stack (Linux operating system, Apache web server, MySQL database and PHP scripting language). The research has identified within the sample 370 deployments of open source solutions by local authorities. Open source software already plays an important role in many councils' IT infrastructure. However the clear tendency is toward councils operating a 'mixed economy' of both open source and proprietary software. Comments from respondents showed that where open source operating systems are used, implementations are generally for dedicated systems rather than office desktop PC environments.

In Figure 20 the current state of open source adoption is presented, according to this survey.

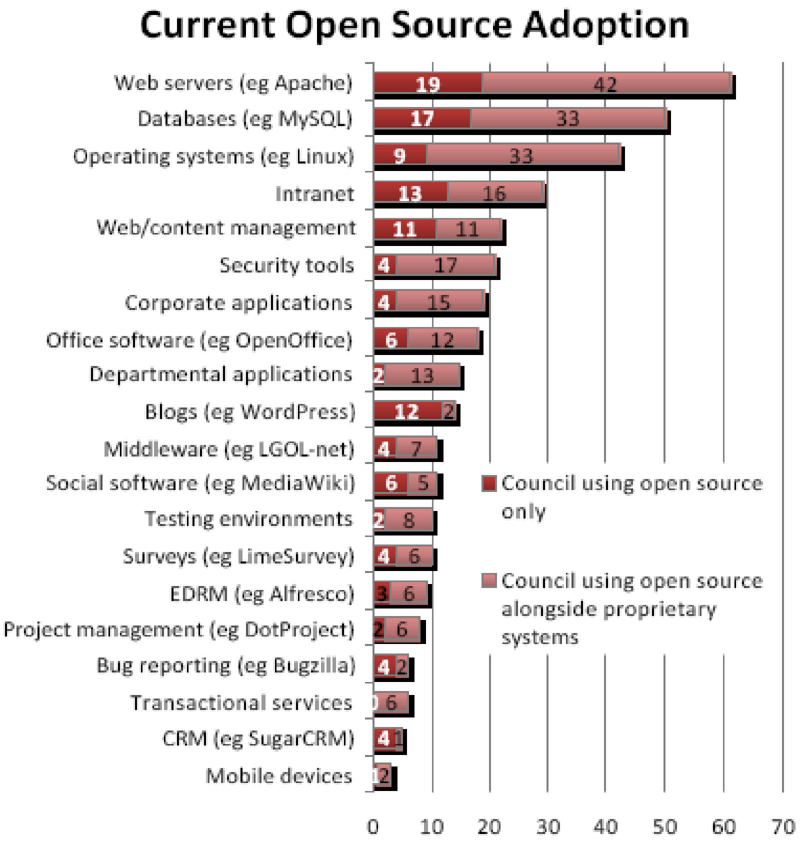


Figure 20: Current Open Source adoption

Future Growth Areas for Open Source in Local Government

Respondents were asked to tell to the researchers what they thought would be the top three software areas where open source will have the greatest impact in local government over the next two years. Overwhelming, respondents pointed to office software, followed by operating systems, web servers and databases.

Also featuring highly were web publishing, content management and social software, such as collaboration platforms and Wiki technologies, reflecting the rapidly increasingly use of Web 2.0 platforms in local government.

Moreover, in Figure 21 the sectors where open source software will impact in the future are presented.

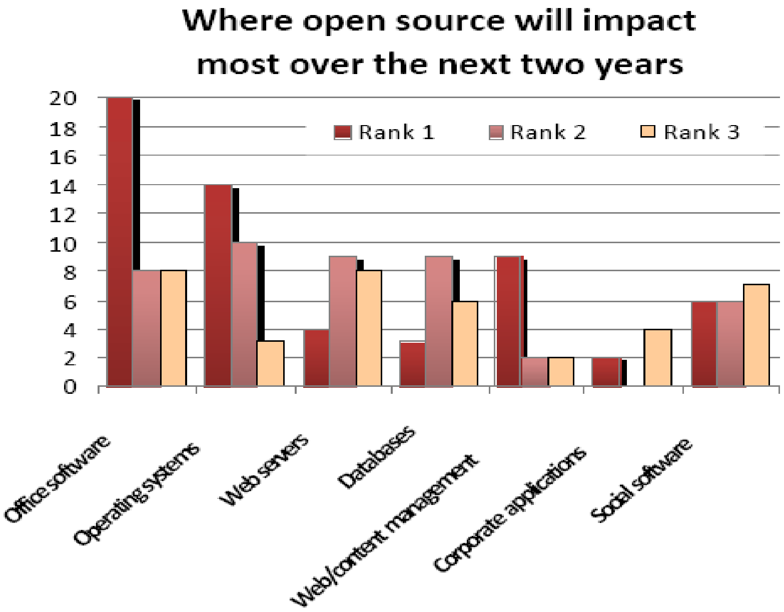


Figure 21: Where Open Source will impact most over the next two years

Councils' Thoughts on Software Licensing Costs

The survey asked local government members to specify, roughly, how much of their council's IT budget was apportioned to software licence fees. In over half of the responses, software licensing costs commonly represented a substantial proportion of between 30% to 40% of local authority IT budgets.

Of those who answered this question, 50% felt this was too high - none believed this figure was too low. Elsewhere in the survey, approximately three-quarters of respondents (76%) said their council was sometimes 'too reliant on proprietary software suppliers', with 30% agreeing strongly with this statement (Figure 22).

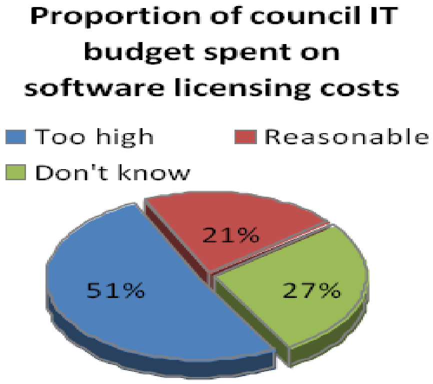


Figure 22: Proportion of council IT budget spent on software licensing costs.

Open Source Benefits in Local Government

Given the need for councils to control IT and business costs, the survey confirmed most local government users of open source are primarily attracted by the lower costs of the software. Councils as a whole see the cost savings benefit as the key advantage of open source and as such, this aspect is valued very highly by local authorities. Cost savings emerged, by a clear length, as the top reason why local authorities had chosen open solution solutions. Three quarters of respondents (75%) cited lower cost as one of the most important factors in their decision to use open source. Following this, as the particular suppliers. Almost half (47%) of those surveyed cited this reason, while 41% were drawn by the functionality of the software itself (Figure 23).

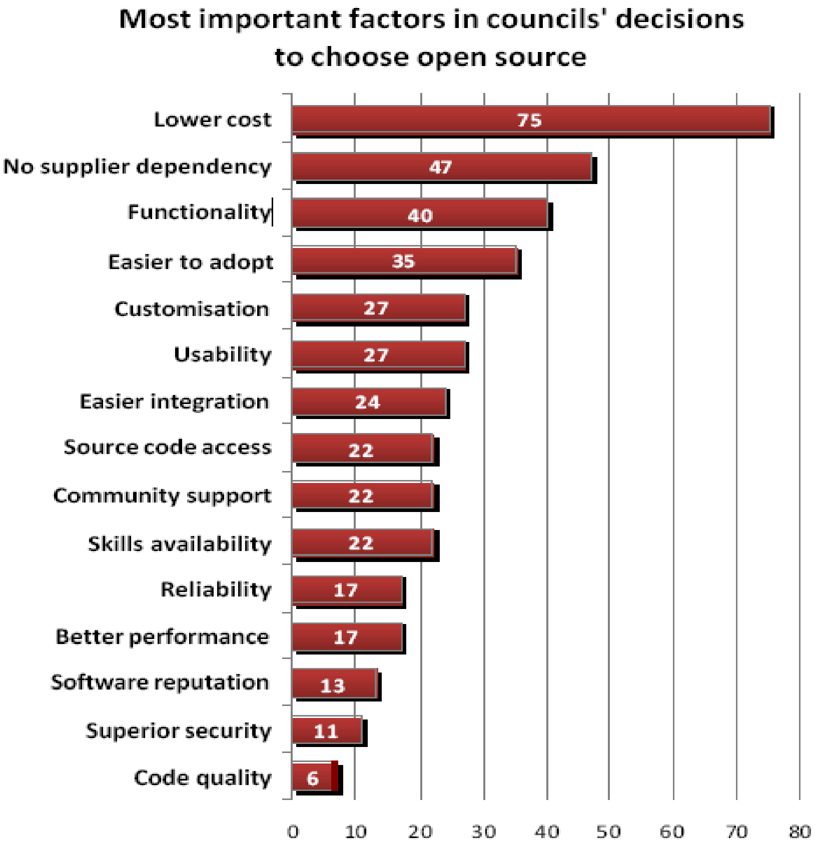


Figure 23: Open Source Benefits in Local Government

Some 34% were attracted by the prospect that an open source solution would be less difficult to adopt than a proprietary one. A similar percentage (27%) cited the customisation and usability aspects of open source as an important driver, while 24% put forward the ease of integration. Less important reasons for choosing open source included access to source code and community support, along with the availability of skills to implement the solution, rated by 22%. Fewer still cited reliability, security or the performance of the software as a main factor in their decision, is the view that open source affords councils the potential freedom from dependency.

This generates the question which are more fundamental concerns for councils than the actual track record of the software in question that when choosing software: cost efficiencies or freedom of vendor-lock-in.

Barriers in Local Government

Overall, the majority of respondents (63%) felt the benefits of open source software generally outweighed the drawbacks. This reflected a trend throughout the survey whereby at least a third of all respondents from local government appear yet to be completely convinced by the arguments of open source proponents. There are, however, clearly major barriers to entry in local government. As the survey found, only a quarter (24%) of respondents said their council considered open source when procuring software – with 44% stating this was not the case. We therefore sought to pinpoint what these were and which were seen to be presenting the greatest challenges (Figure 24).

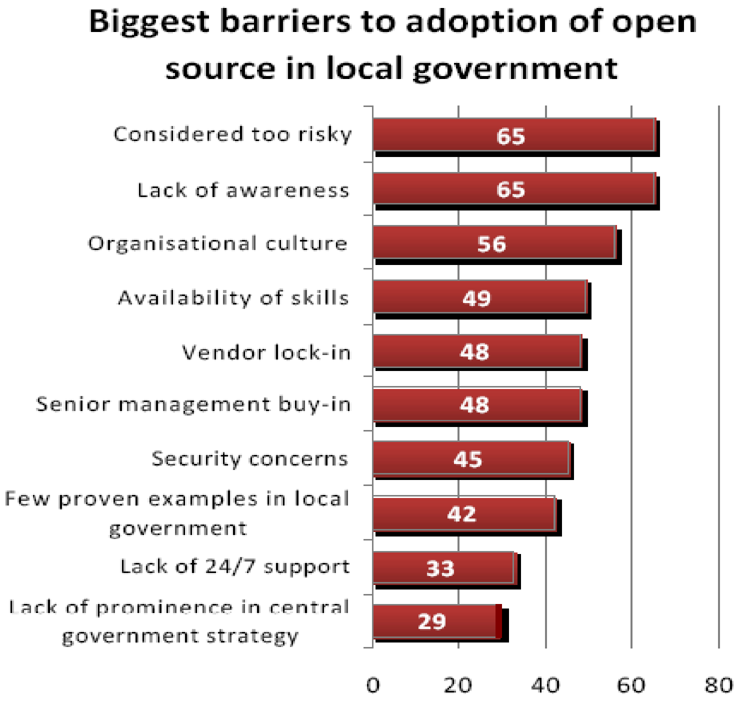


Figure 24: Barriers to adopt open source

Of the 168 respondents, 61% registered specific barriers with responses, indicating a multitude of important factors (Figure 24). Here there were both some surprises and some very predictable views expressed. Technical issues were felt by councils to be a significant barrier to open source adoption, but not the most important. As the responses showed, primary obstacles identified by the respondents tended to relate overwhelmingly to organisational culture issues in local government, such as lack of buy-in or understanding at senior level, resistance to change and internal objections from management and users. Such sentiments were reflected in the comments from our respondents, such as these two examples below:

'*We've used open source in the past. In general users were hostile as they merely see it as the organisation attempting to cut corners by not giving them 'proper' (ie Microsoft) tool set. As an IT manager I also feel there is a degree of risk as you are at the mercy of technical evangelists who can up sticks and leave you without support*'.

'*Senior IT Management seem to be locked into thinking that only expensive, badly - integrated platforms incurring massive and complicated installation projects are worth considering. 'It must be good because they charge so much!*'

The perceived risk of open source continues – rightly or wrongly – to be an extremely serious concern for councils, with strong indications that this is impacting very negatively on its adoption in local government. Around two-thirds (65%) believed the perception by management that open source is 'too risky' is preventing councils from increasing their using open source.

A similar percentage felt that take-up by local authorities was also mainly held back by a lack of awareness of open source software, with 42% also citing the scarcity of proven examples of open source use in local government, which may be linked issues. When asked what would help councils to increase their adoption of open source, the top issue highlighted was the need for more visibility of successful implementation across local government, in particular high-profile, practical case studies of success, proof of concepts and greater knowledge transfer of 'what works'. Many held the view that if councils could see successful, real -world examples in leading authorities, many would be more inclined to follow suit. One respondent set out their wish list of what they required:

'*Good, relevant examples of IT infrastructures based on open source. Five year cost comparisons. Value added exercises (what you actually get and what you use, out of what you have procured, after one year). People to talk to and who can come and persuade the intransigents. A persuasive argument for the simplicity of open source, when considered at Enterprise level or at a departmental basis*'*. A persuasive argument of three-year efficiencies. But mostly people to talk to who are knowledgeable and experienced.*'

Concerns about security, being 'locked' into an existing commercial supplier and the willingness of vendors to integrate with open source applications also featured prominently here, whilst other hurdles mentioned included availability of skills and support. Notably, central government's historical lack of encouragement and support for open source, and its perceived unwillingness to 'lead by example', were also identified as a significant factor in councils' ongoing reluctance to adopt open source. 'The Cabinet Office promote open source software, but hardly use any themselves', said one response. Other respondents felt a stronger lead from Whitehall 'in the same way they pushed the IEG initiatives' would give councils the necessary impetus to explore alternatives to proprietary software.

The need for a central government-'endorsed' list of open source applications was also mentioned. One respondent asked in particular for 'guidance from central government that we can wave at IMT who are too concerned by risks - and include benefits as well so we can get buy-in from senior management.' One respondent articulated the key issues at stake in this report in the following comment:

' *A big factor in pushing open source use into local authorities would be larger backing from central government; we are in a climate where we are trying to save money and massive savings could be made from bringing in open source alternative solutions into the workplace, dramatically reducing licensing costs. Some kind of centralised,government-driven support service (or even one distributed by a group of specialised officers working for different local authorities) could help to reassure local authorities that this could help their budgets and that there would be (albeit basic)* *support available to them. I also think there is a certain psychology in using 'branded' products such as Microsoft Office instead of OpenOffice, as it is a tried and tested name. If these boundaries could be overcome and it could be prove that the open source alternatives do just as good a job, then open source could play a big part in our government organisations.'*

2.1.6 Criteria for adopting open source software in Public Administrations ([9])

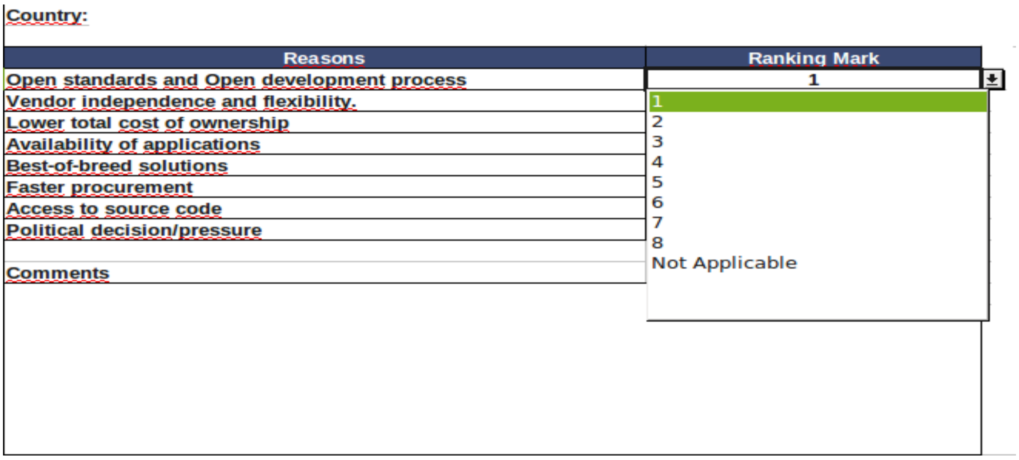
Here the researchers analyse the most important criteria taken into account by the public administrations of thirty countries when adopting or choosing open source software, within the limitations inherent in this type of study.

It is also intended to be a valuable contribution to the future of the open source software sector, due mainly to the fact that it is the result of collaboration between members of many public administrations worldwide. 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 this dossier.

The first stage of the study, a review of the literature published by CENATIC observatory, has identified eight different main criteria taken into account by public administrations when choosing to use open source software.

- Open standards and open development process
- Vendor independence and flexibility
- Domestic economy.
- Low total cost of ownership
- Availability of applications
- Best-of-breed solutions
- Faster procurement
- Access to source code
- Political decisions and initiatives

CENATIC has created an evaluation sheet based on these 8 main criteria that takes into account the weight assigned to each of them, based on the opinions of the participants. The weight given to each criterion ranges from 1 to 8, with 1 representing the least importance and 8 the greatest importance (Figure 25).



Country:

Reasons	Ranking Mark
Open standards and Open development process	1
Vendor independence and flexibility.	
Lower total cost of ownership	
Availability of applications	
Best-of-breed solutions	
Faster procurement	
Access to source code	
Political decision/pressure	
Comments	

Figure 25: Evaluation sheet. Source: ONSFA. 2011

Data was collected by means of an evaluation sheet sent to related professionals of the public administration in the participating countries. The objective was to obtain first-hand information on the criteria considered when adopting open source software, based on the opinions of these professionals. The adoption criteria for open source software are quite different and are used for different purposes depending on the country. These criteria, identified by CENATIC for use by the public administrations when making decisions to adopt open source software and the obtained results, are described in more detail below.

Summary of the analysis

This analysis identifies the most important criteria (Figure 26) taken into account when deciding whether or not to adopt open source software in the different geographical areas for use by the Public Administration. In terms of the criteria for adopting open source software, the dossier concludes that the administrations are influenced by criteria such as TCO, with criteria such as vendor independence and flexibility, open standards and open development process being particularly important. However, the public administrations are less easily persuaded by criteria such as faster procurement, best-of-breed solutions and political decisions and initiatives.

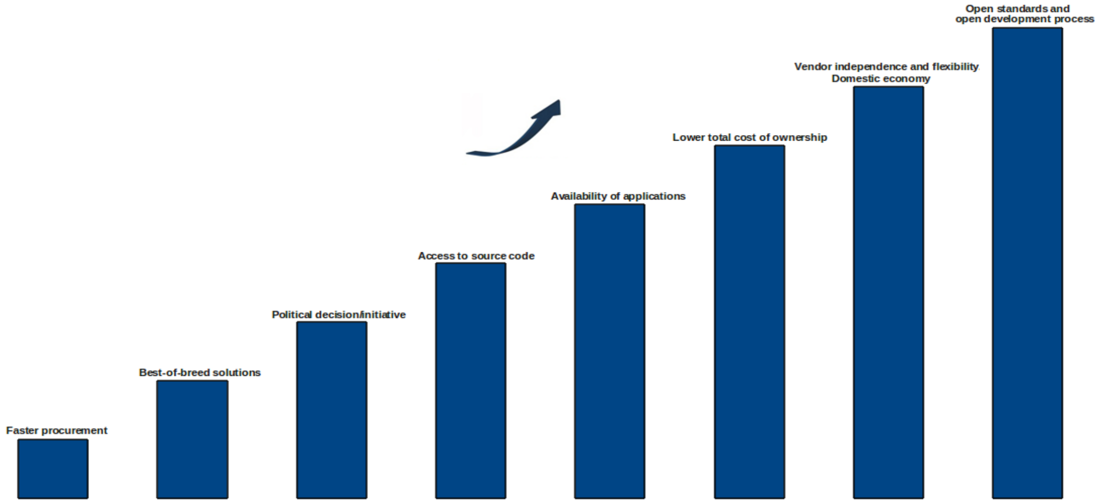


Figure 26: Overview of the situation. Source: ONSFA. 2011

Open standards and Open development process

Generally speaking, open source software products usually follow open public guidelines and specifications. The use of open standards favours systems interoperability of, as well as the development of new services and content. These are essential considerations when implementing e-Administration, and particularly when ensuring that the services provided are accessible to everyone. In this regard, the use of open standards ensures the participation of the different parties involved in these administrations, promoting an open software development model. Considering only each country's score for "Open standards and Open development process," it can be seen that 57% of the countries consider this criterion to be very important and important. Spain forms part of this group of countries. Figure 27 presents which ranks the process of many countries in open standards and development processes.

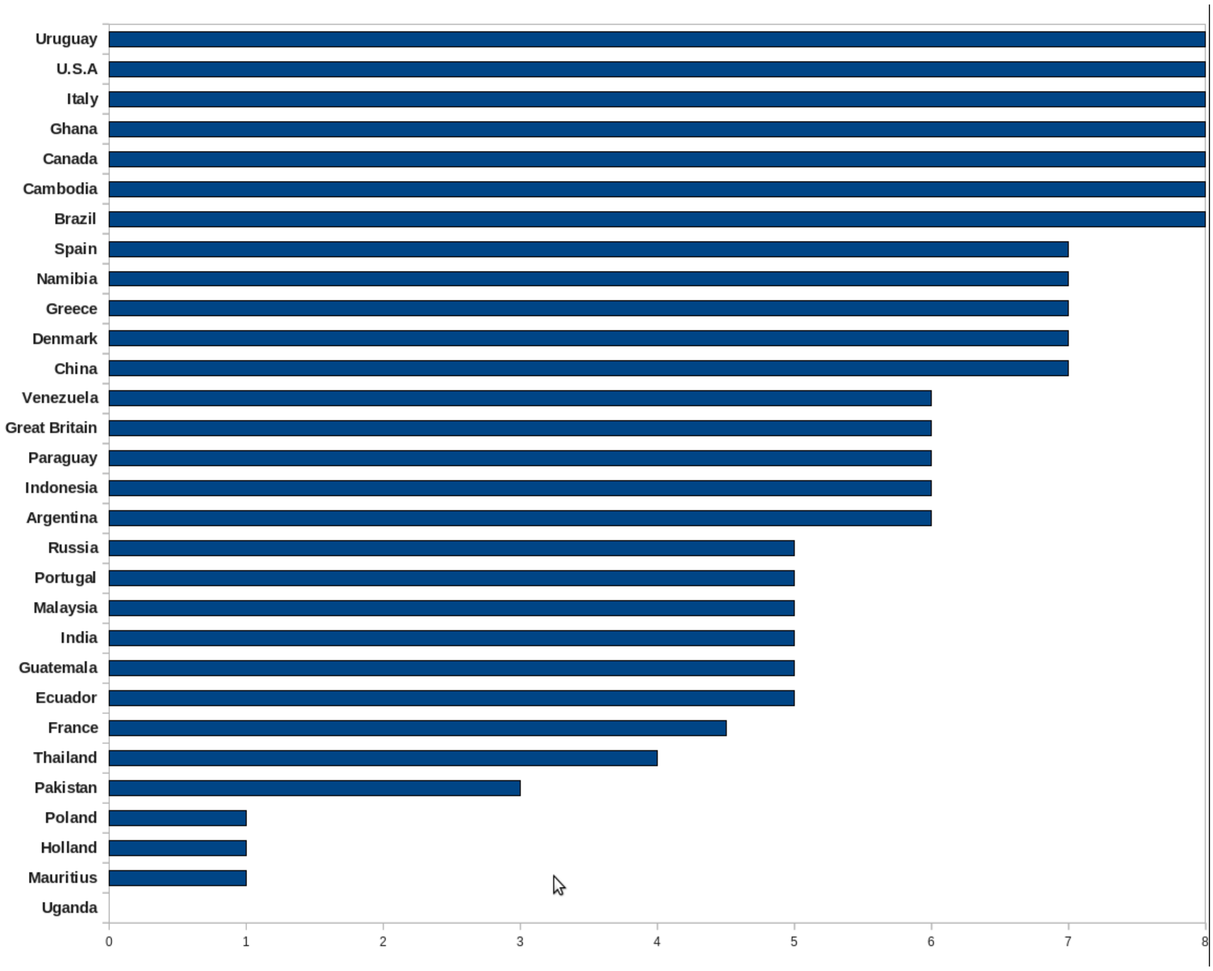


Figure 27: "Open standards and Open development process" ranking.

Vendor independence and flexibility. Domestic economy.

The fact that open source software is based on open standards and regulations means that vendors can be changed at any time, so customers can choose the best solution in terms of cost and the features provided. Open source software therefore helps reduce dependence on multinational software companies and promotes the national ICT sector, enabling smaller companies to compete and benefit from the increased opportunity to do business with public administrations.

Considering only the scores obtained for "Vendor independence and flexibility. Domestic economy" we can see that 47% of the countries consider this criterion to be very important and important (Figure 28).

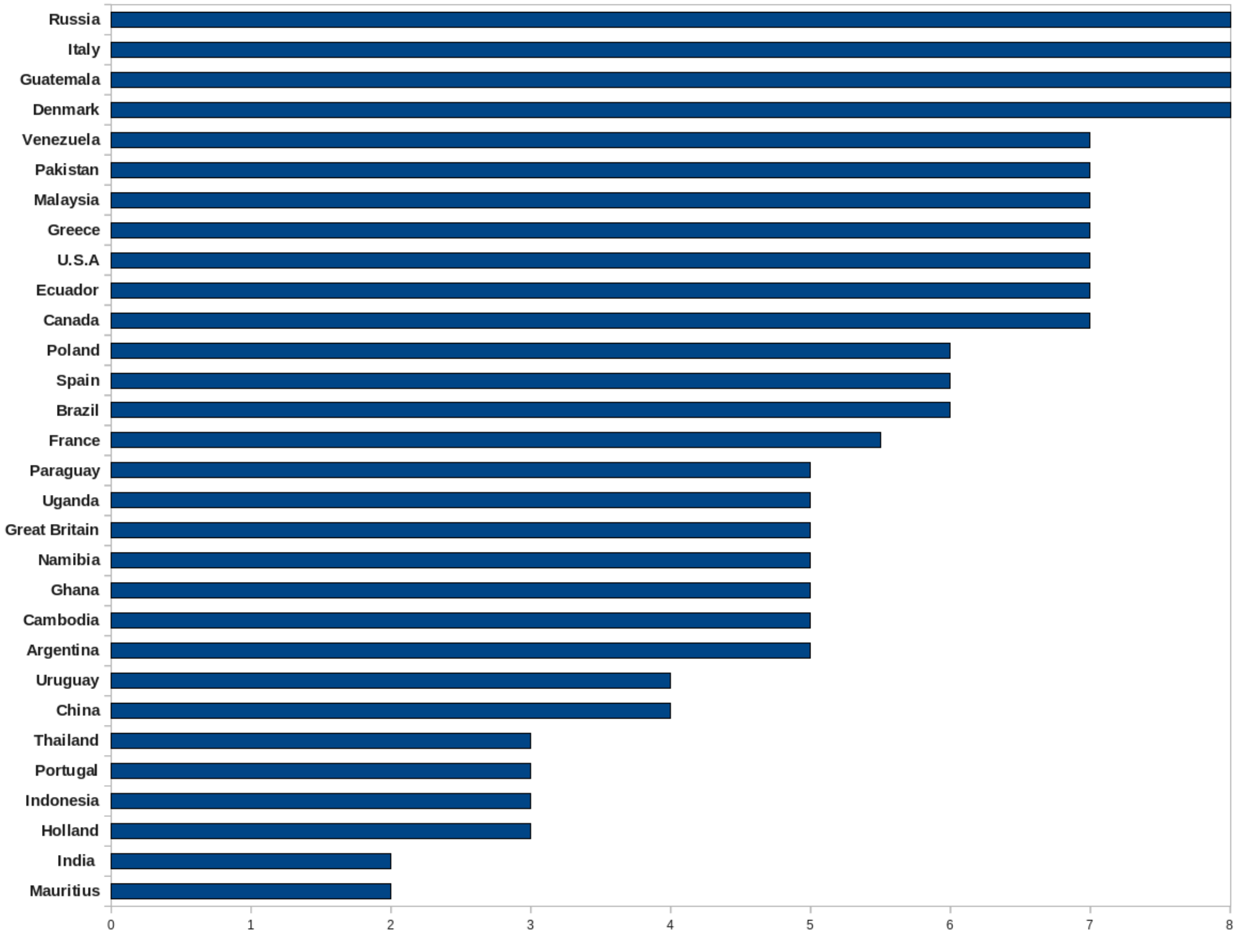


Figure 28: “Vendor independence and flexibility. Domestic economy.” ranking.

Low total cost of ownership

Widespread use of open source software reduces costs by taking advantage of economies of scale and the reuse of code. It is recommended that software procurement or migrations be carried out independently of acquired licenses. It is crucial to assess the Total Cost of Ownership, including all relevant factors. The initial purchase price is easily measured, but this is just one of the many factors to calculate the TCO. User training, maintenance needs, updates and support must also be taken into consideration.

Considering only each country’s score for “Low total ownership cost," we can see in Figure 29 that 50% of the countries consider this criterion to be very important and important.

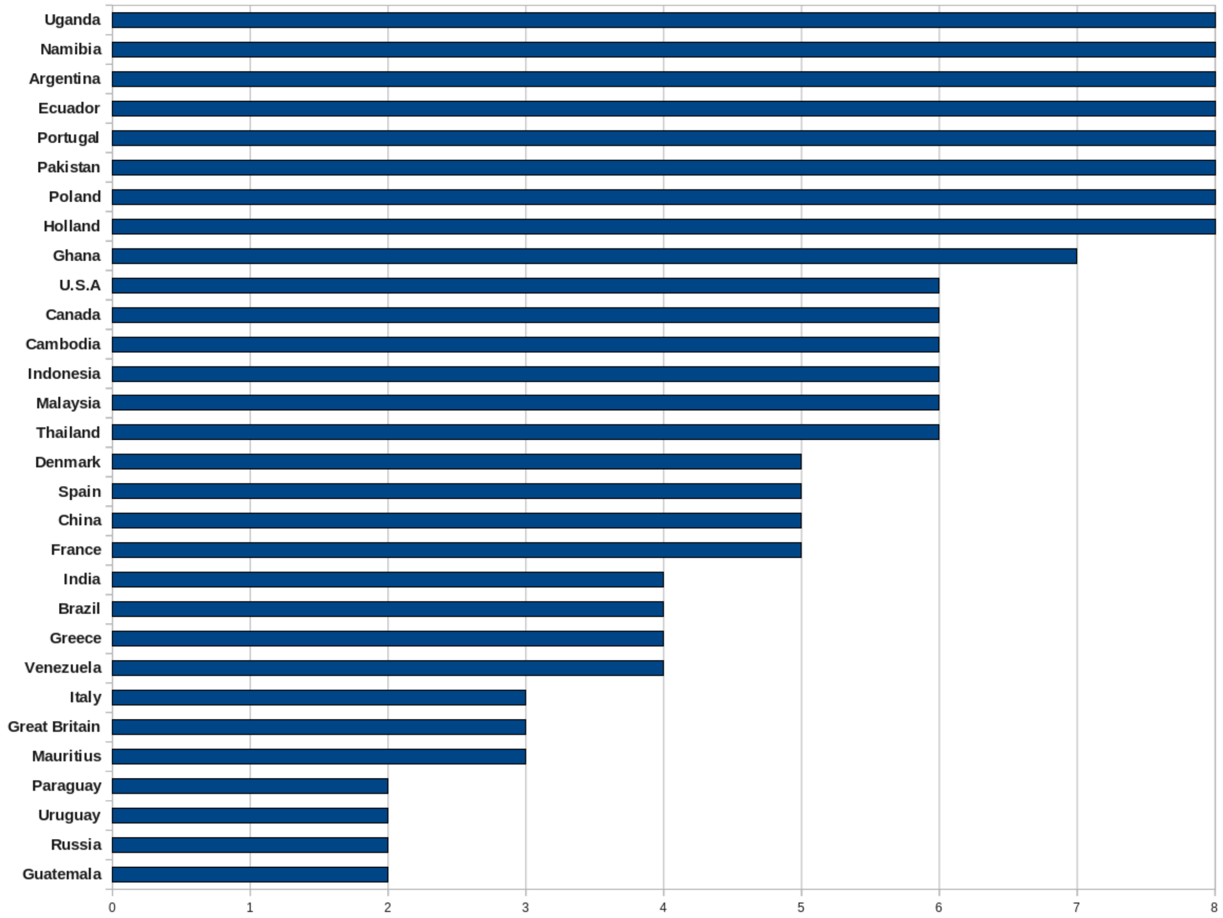


Figure 29: "Low total cost of ownership" ranking.

Availability of applications

There is a large number of programs and a great deal of support in the open source software sector. Commercial companies would need to dedicate several million Euros and thousands of people each year to produce the equivalent of the open source software currently available. This availability is one of the great advantages of open source software, which, as a general rule, takes advantage of the work that has already been done to develop new solutions, eliminating the need to constantly reinvent the wheel. Considering only each country's score for "Availability of applications," (Figure 30) it can be seen that approximately 43% of the countries consider this criterion to be very important and important.

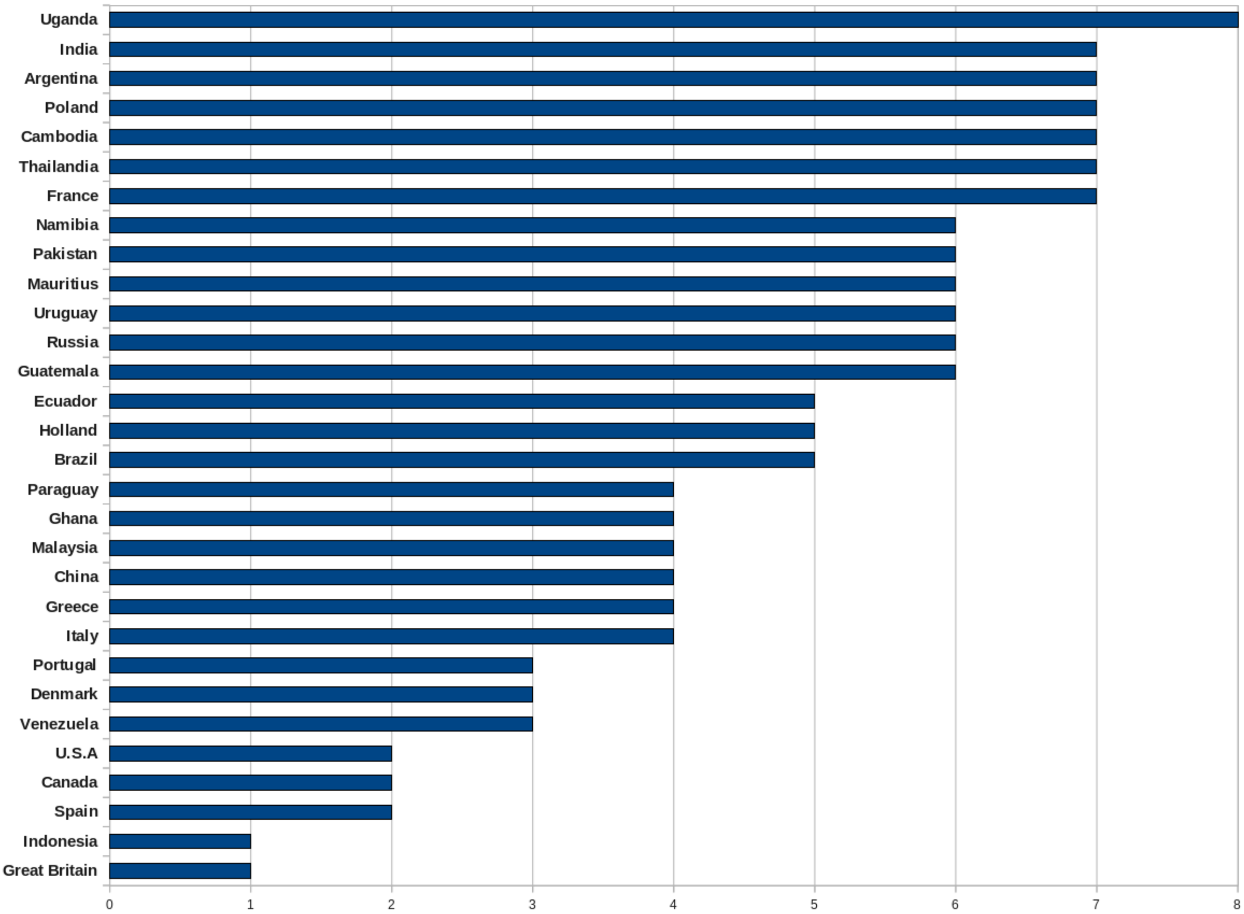


Figure 30: "Availability of applications" ranking.

Access to source code

Access to an information system's code means that its computer programs can be modified in order to improve them, adapt them to our needs and subsequently distribute these adaptations. This free access to programming code makes it possible to adapt the programs, modify them and progressively debug them without having to rely on the exclusive support of a single vendor. Likewise, it provides the opportunity to solve any possible errors or security flaws faster and it facilitates the tasks associated with translating a product into other linguistic modes.

Considering only each country's score for "Access to open code," (Figure 31) it can be seen that 27% of the countries consider this criterion to be very important and important.

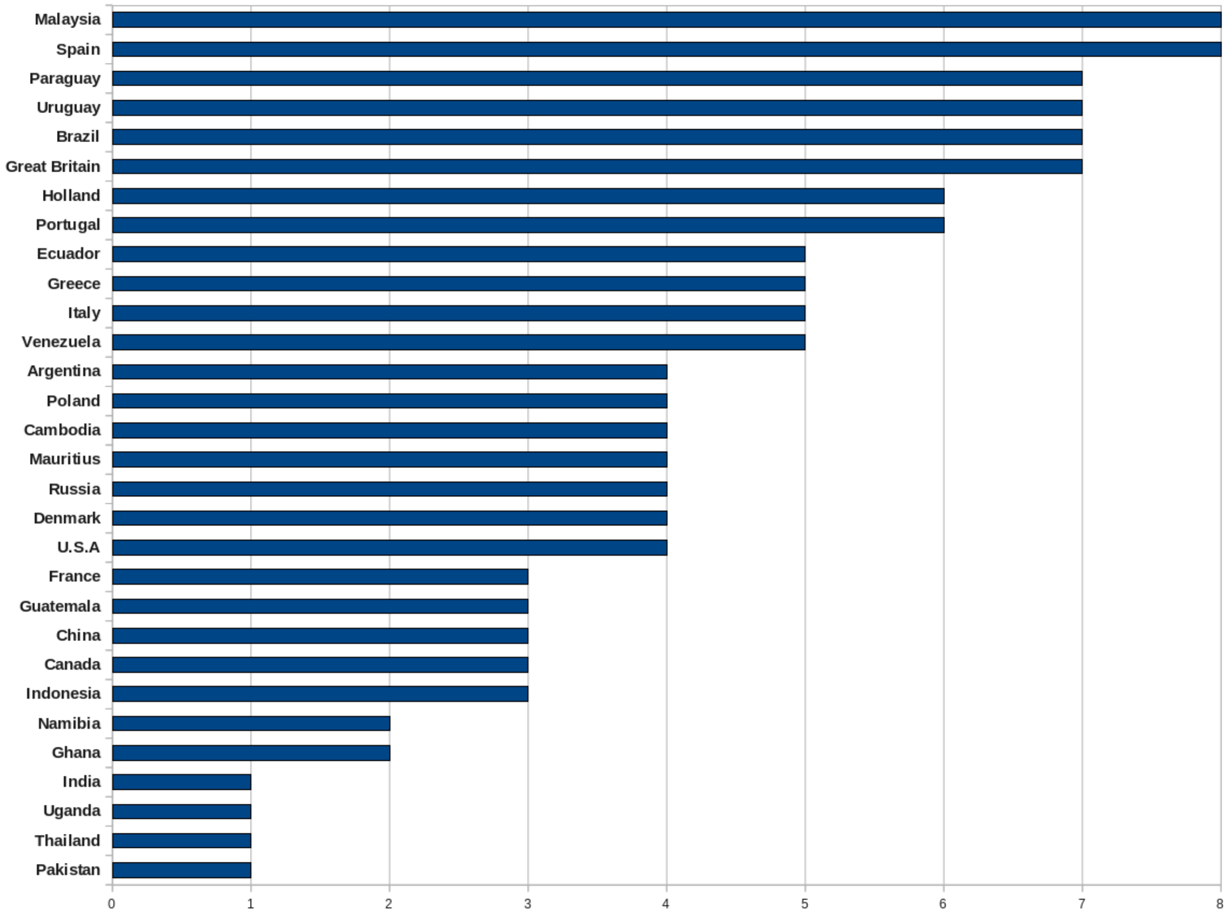


Figure 31: "Access to source code" ranking.

Political decisions and initiatives

Strong support from different Public Administrations is evident, to the point that it has occasionally been incorporated into the framework of the institutions they govern. As a result of this support, administrations have developed an ecosystem of people related to open source software at all levels, including political, administrative, business, social, etc. This powerful ecosystem clearly favours project development based on open source software, as well as the creation of projects and initiatives with a social and participative approach to technological development itself.

Considering only each country's score for "Political decisions or initiatives," (Figure 32) we can see that 37% of the countries consider this criterion to be very important and important.

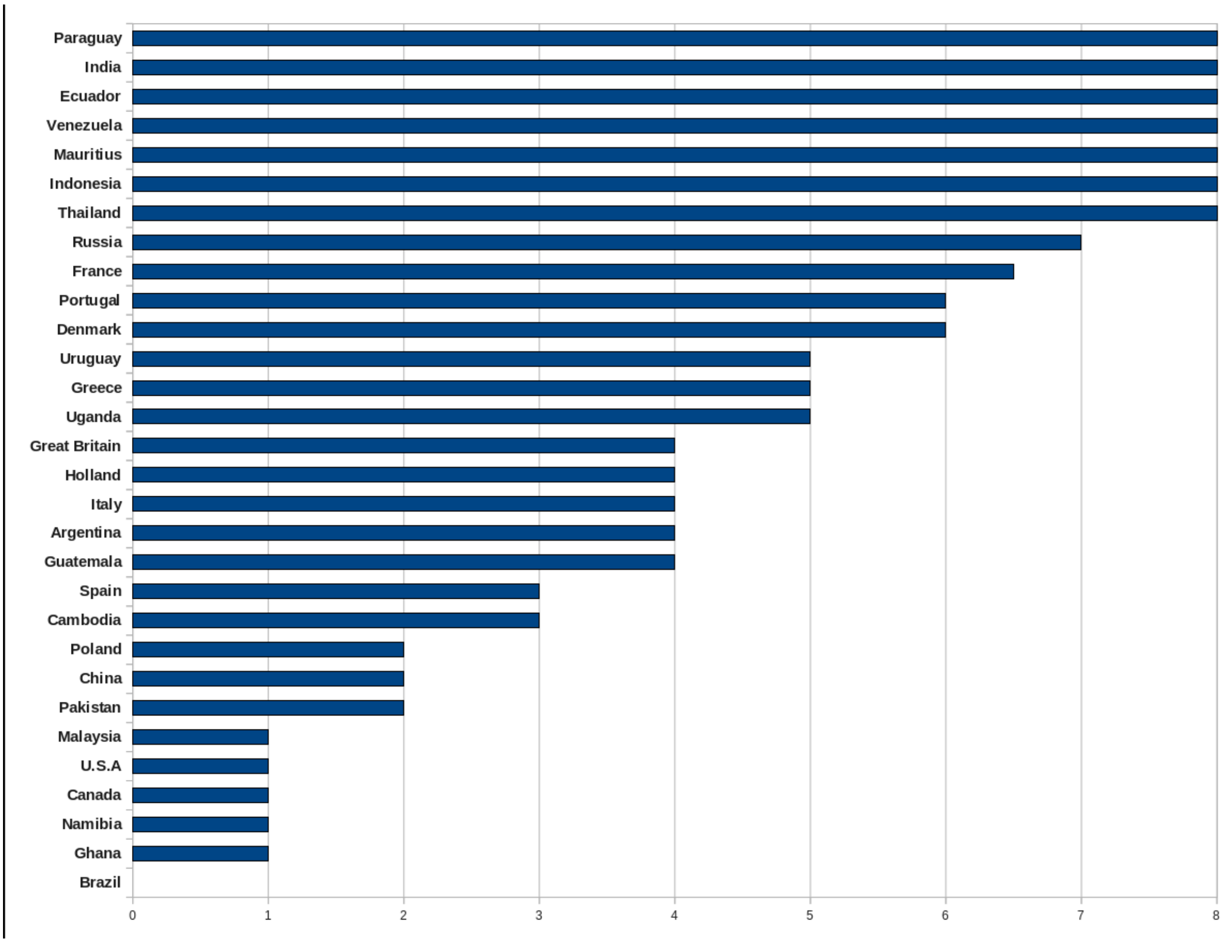


Figure 32: "Political decisions and initiatives" ranking.

Best-of-breed solutions

It is important for Public Administrations to reap the benefits of the adaptations, contributions and improvements they have contributed to (either directly or through outsourcing) as part of the development of the products they use. This improves the quality of the product, enhancing its sustainability as a project and facilitating the adoption of these changes by major manufacturers. These changes are then reflected in future versions, accompanied by a subsequent reduction in the budget needed to integrate and adapt these new versions.

Considering only each country's score for "Best-of-breed solutions" (Figure 33) we can see that 23% of the countries consider this criterion to be very important and important.

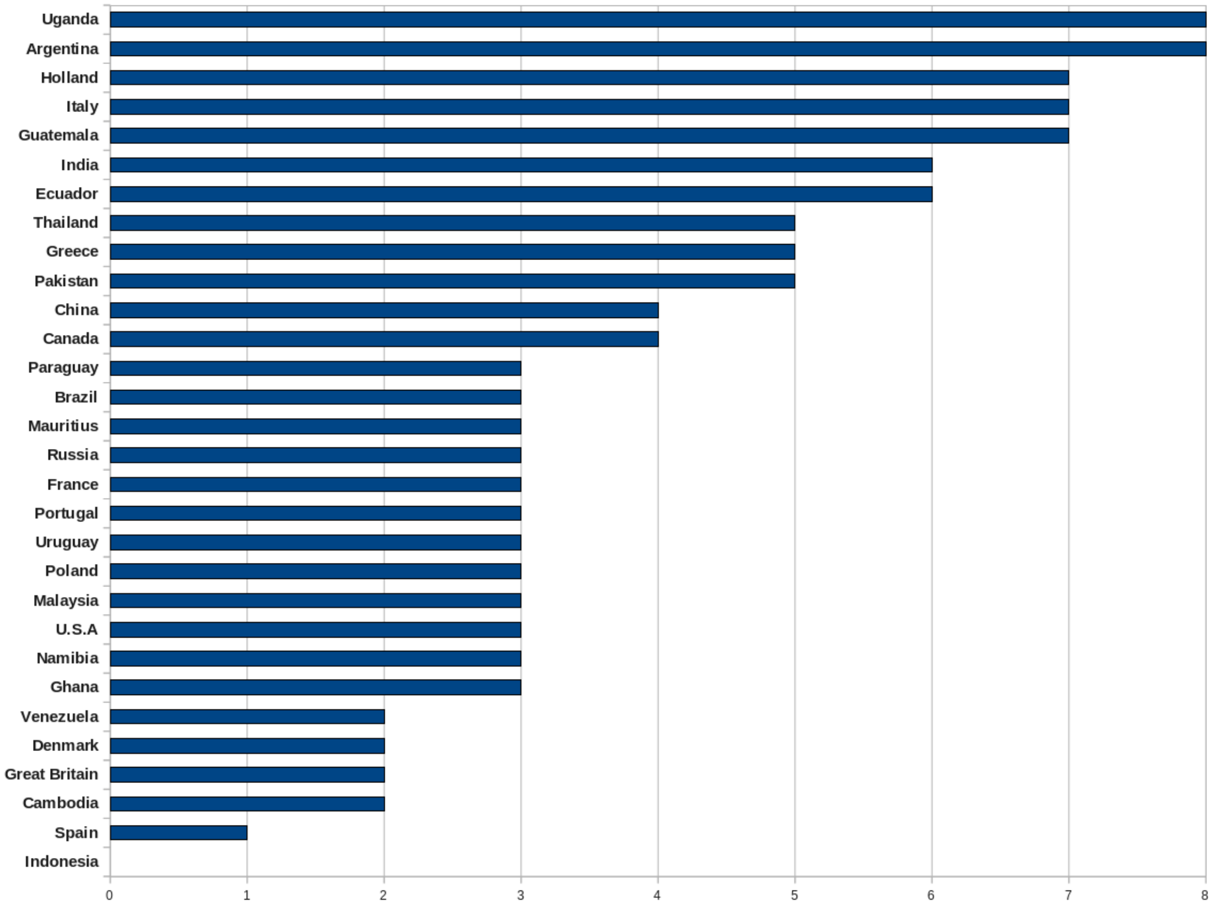


Figure 33: “Best-of-breed solutions” ranking.

Faster procurement

As set out in the legal analysis of the administrative guidelines in the Netherlands¹, the procurement of open source software does not necessarily imply the need for a tender process. This applies in specific situations, for example when software can be acquired free of charge, meaning that not only the licenses are free, but so are the manuals, support and services. If paid services and technical support are necessary, they may be obtained through a separate tender process [44]

Considering only each country’s score for “Faster acquisition," (Figure 34) we can see that 10% of the countries consider this criterion to be very important and important. It should be pointed out that only two countries view "fast procurement" as important, and nearly 50% of the countries consider it to be the least important criterion.

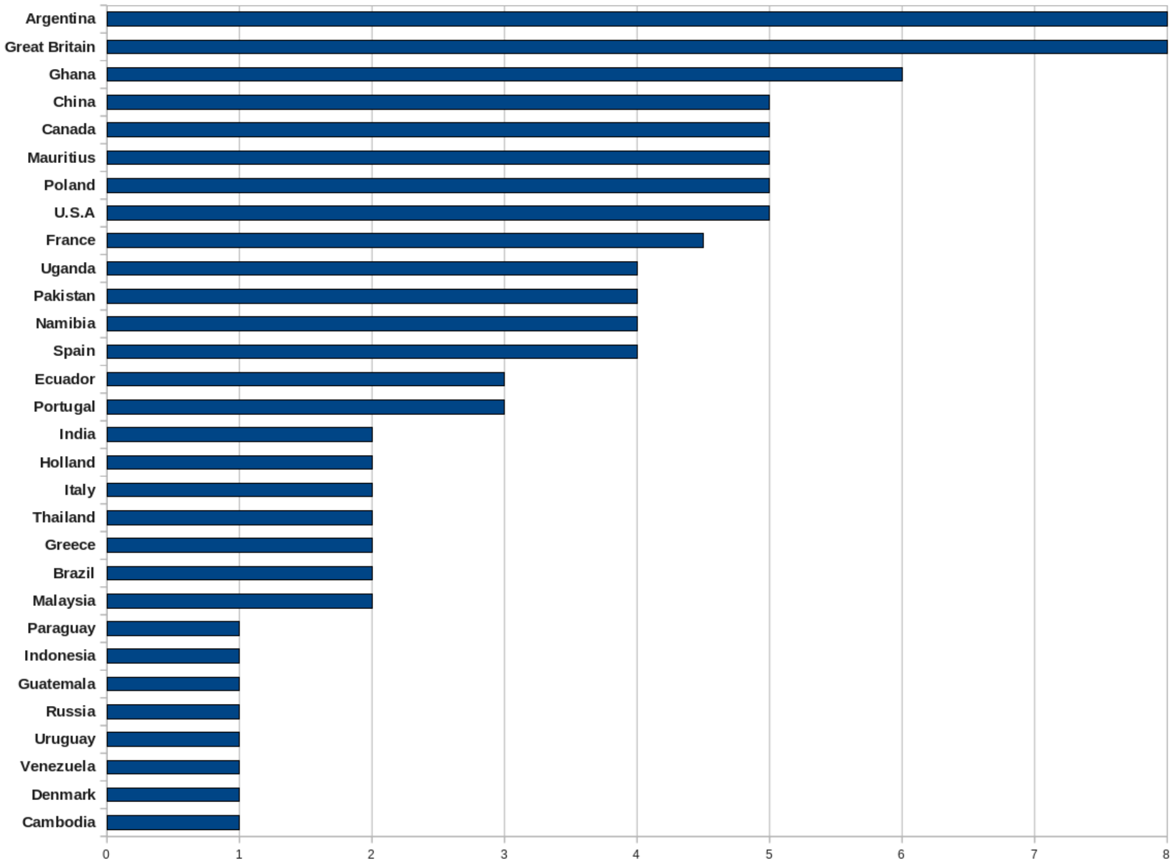


Figure 34: "Faster procurement" ranking.

2.1.7 Danish Board of Technology Open-source software in e-government ([10])

Analysis and recommendations drawn up by a working group under the Danish Board of Technology - The socio - economic consequences of open-source software

Open-source software has been the subject of a widespread myth, to the effect that removing the licence fee – which only accounts for a small part of total IT costs – does not make up for significant uncertainty on quality, performance and maintenance costs in open-source compared with proprietary software. All that remains then is to quantify the possible differences between open-source and proprietary software using a socio-economic yardstick.

The purpose of this section is to illustrate the socio-economic differences between the use of open-source software and proprietary software in public administration in Denmark. A socio-economic analysis assesses the total loss that follows from decisions taken against the background of limited information and imperfect market competition. It should be emphasised at the outset that there is no information as yet, in available statistics, on even the most basic use of IT in public administration.

Consequently there is no information on how widespread open-source software is in public administration in Denmark, but the general impression is that it is used to a marginal extent. The following calculations are therefore based on rough estimates and specifically reasoned assessments in each individual case. Careful estimates have been used as a rule, so that all the calculations are intended to show possible socio-economic gains or losses under well-defined conditions.

In order to be able to assess the socio-economic consequences of replacing proprietary software with open-source software, a comparative study is made of the particular types of software, using relevant assumptions from the general model for the types of software concerned. This survey is principally concerned with infrastructure and desktop software, because there is a possibility here of extrapolating from previous studies.

Calculations of the socio-economic consequences are based on foreign analytical results, assuming that they can also be generalised to the public sector in Denmark, Danish conditions being involved in the calculations as far as possible. This is a debatable assumption, but the best that can be done in the circumstances, using cost differences from the US, which for many reasons has software markets more open to competition than in Denmark. These cost differences will consequently under-value rather than exaggerate the relative differences between open-source and proprietary software, on the basis of the observation that stronger market competition leads to lower (actual) prices. The following calculations are therefore to be regarded as rough estimates, which indicate some relative size ratios of a more closely specified nature. It is considered that the advantages in stating rough estimates of size ratios outweigh the drawbacks that follow from the uncertainty over the possible socio-economic gains, uncertainty that will always be associated with calculations of this kind, but which should not prevent efforts to form a picture of the economic proportions. These calculations should be included in the strategic considerations that become necessary when considering the volume of the total public investment in IT.

Qualitative socio-economic assessments

Below the researchers put forward a general model for investments in open-source software in different situations of choice. Here the survey examines the socio-economic differences in choosing between open-source and proprietary software in a situation where no investments have yet been made and in a situation where investments have been made in proprietary software, but upgrading has become possible (for example from Windows 97 to Windows XP). The latter Is discussed as a change of software platform, where replacement of the platform used hitherto can take place either by choosing new proprietary software (possibly implying change of hardware), or by choosing open source (possibly keeping existing hardware).

Cost components in new procurement and changing of software

- Price and/or licence fee

  Procurement and/or licence costs.

- User-friendliness: The effect of the user-friendliness of the software on indirect costs in the user environment (long ’response times’, ’deeply buried’ screens and functions, confusing icons or screen instructions etc.)

The working group is not aware of studies that provide evidence of major differences between open-source software and proprietary software with regard to user-friendliness. It is assumed that these costs are a function of the specific design and independent of whether open-source or proprietary software is concerned.

- End-user training: Requirements for education and training of end-users

  The working group is not aware of studies providing evidence of differences between open-source and proprietary software with regard to learning. It is assumed that these costs are a function of the specific design and independent of whether open-source or proprietary software is concerned. Where there is no functionally equivalent open-source alternative to proprietary software, there is additional expenditure on training. One course day including course payment represents a value of 1% of the annual norm of salary.

- Training of IT staff: Requirement for learning in the internal IT maintenance function or for new service contracts with stated, chosen level of service agreement for suppliers

  The requirement for local expertise is generally higher in the case of open-source software than with proprietary software. At the same time, familiarity with open-source software, particularly for desktop software, is less than with the most widely used proprietary software. It is assumed that the build-up of skills will be greater for open-source software than for proprietary software, which puts the emphasis on the supplier retaining control over its software, in contrast to open source. Switch to open source will normally be accompanied by requirements for courses, the extent of which will depend on prerequisites for skills. A switch to a new, upgraded version of proprietary software is generally accompanied by a need for continuing training. Difference in scope and price of continuing training rules out a clear conclusion in choosing between open-source and proprietary software. Both cases will imply using several per cent of the annual norm for salary costs in the own IT department or for hiring external consultants.

- Software compatibility: Compatibility of surrounding software and prerequisites for network interoperability and ensuring this.

  Infrastructure and desktop open-source software can be used on almost all platforms. Desktop open source has a smaller installed base, so that integration with the surrounding environment has been less thoroughly tested. For open source there is in principle better opportunity to integrate with surrounding software if this is also open-source, while the program interfaces of the proprietary software may limit the scope for integration

When the researchers speak of irreversibility, it also reflects the fact that previous investments, for example in skill-developing certification courses, lose their value (or most of it) on the change-over to another platform, when skills are relatively strongly supplier-specific with a limited possibility of re-use. Requiring both new investment in skills and the writing-off of investments made to date in building up skills makes chosen proprietary software a stronger tie for cost reasons than that related to the advantage of reduced licence cost.

Irreversibility also follows with a shift in basic formats (e.g. file formats for office software), where a dominant supplier attempts to move its market to a new 'standard' to weaken new and smaller competitors. Conversion is made more difficult until new tools are developed, after which conversion expenditure makes a further contribution to making it difficult for an alternative platform to be maintained.

Control of widespread formats, for example, is used as a strong competition-guiding instrument, albeit always with risks of negative customer response, since any productivity benefits to the customers with the new format are accompanied by an increased tie to the supplier. As only the supplier of a dominant software product can implement changes of format, the supplier concerned decides what changes take place and when. These decisions are taken on the basis of the supplier's assessment of the competitive terms.

Competitive conditions are a function of the software decisions the market takes. If the market takes a decision on a short-term basis, it is without an assessment of the circumstances mentioned here. If an intermediate perspective is applied, the assessment of software investment may include said circumstances. If the market is characterised by many small decision-makers, it is difficult to escape the advantages of doing the same as everyone else (accepting dominance), while the decision of a larger group on a collective switch will quickly have a visible effect on the market, as the dominant supplier will be compelled to respond.

Factors for assessing irreversibility of investment in software products

- Specificity: Any program imposes a set of requirements for its hardware and software environment

  Choosing software entails a degree of lock-in, i.e. incontrovertible loss on switching to alternatives which is greater the more specific software is in its requirements for its environment.

  Proprietary software by nature is likely to increase the requirements for the use of specific software from the same supplier. Open source allows increased investments in software, as this software can normally be used in many environments (this may possibly not be applicable to custom built software). Equivalent broad use is aimed for with international standards for proprietary software

- User learning curve: Software is knowledge-based product with a learning time for users

  Loss in training costs is accompanied by indirect losses in reduced production during training time compared with the case of full experience Open source has less discontinuity for users than proprietary, where there is an incentive to make software 'obsolete' to increase the market

- Compatibility: How capable a software product is of working together with other software

  Complementarity effect (positive economic value) and the opposite (if there is a requirement for conversion routines etc.) Proprietary software is normally backward-compatible but rarely forward-compatible precisely so that a new market is created. Proprietary software is more strongly integration-oriented in order to increase the market. But at the same time problems are created for third parties. The lock-in effect is increased.

- Support learning curve: Maintenance and prerequisites for support - Skills of IT department in supporting users and maintaining software

  Specific investment in staff (learning curve) and software tools, which cannot be fully re-used, entails loss on switching software. The costs in switching are a reason why suppliers of dominant software can charge a premium price compared with the smaller suppliers. Proprietary software is dependent on courses that give precisely the insight to administer their product. Open source does not have any precise demarcation of what is ’sufficient’ as it depends on the level of aspiration among the operating personnel of the organisation. Alternatively consultants may be used.

- Integrability: Any software environment needs to effectively integrate new software, and software products can do this more or less effectively

  Software integration costs increase in line with the incompatibility of software products and increase the barriers to acquisition of new software. Open standards reduce the barriers (all other things being equal). Proprietary software defines interface on the basis of competitive terms, open source defines them on software technology grounds.

Quantitative socio-economic assessments

Quantitative models make great demands in relation to data collection, as is the case with the models used by consultancies in comparisons between alternative products. The working group has not gathered data from all public authorities, or even a selection of them, to illustrate socio-economic alternatives.

The difference between open-source software and proprietary software is not limited to the licence price. A consultant report [5] shows that Windows server software (using IIS as web server) requires significantly more maintenance than both Linux and UNIX server software (which both use open-source Apache web servers). The report finds, for an approximately standardised analysis of ’total cost of ownership’ in a selection of 14 enterprises, that UNIX (Solaris) is 7.5 times and Windows 2.5 times more expensive than Linux over a three-year period. The study has not been able to quantify breakdown as a result of virus attack or rebooting of Windows servers following the installation of new ’patches’, although this is a relatively common maintenance task that distinguishes Linux and UNIX from Windows, in that they do not require rebooting. We take as basis below a study that finds that TCO differences between Linux and UNIX of the order of 1.8 to 5.5 for UNIX depending on the set of tasks.

A Gartner' s report [22] emphasises the importance of building up local skills in IT support and the need for consultancy agreements for open-source products as well as differences in the nature of tasks in the assessment of the overall economic advantages between proprietary UNIX, Windows and open source, which confirms the relevance of our economic model. On the other hand, neither of the reports attempts to make a complete assessment of the significance of irreversibility other than highlighting some of these factors as a cause of a relatively smaller advantage in switching to open-source infrastructure software.

It must be emphasised that the chosen examples do not cover absolutely any open-source or proprietary software. The researchers have chosen to base their analyses on software in very widespread use. They cannot claim on the existing basis that open-source software will always be more advantageous than proprietary software, firstly because there is no documentary evidence that design methodology and support for open-source software necessarily provide better-quality software (containing fewer bugs, greater user-friendliness, easier integration etc.), and secondly because there is no evidence that it results in quicker software development than other methodologies.

The studies undertaken in this report provide evidence for the chosen examples that substantial economic advantages have been observed for open source in desktop software (office programs and operating systems) and in server operating systems. The researchers have presented this result while being fully aware that they are not aware of all the conditions needed for them to be able to generalise. This reservation is mentioned because reports from suppliers of software are subject to equivalent problems in establishing all the conditions that need to be met for it to be possible to generalise from study results. This survey has refrained from assessing the productivity differences among software, as these assessments are particularly sensitive to local organisational circumstances.

It has not been possible to calculate what proportion of the total software expenditure the selected type of software represents in the public sector in Denmark. The selected software products are characterised by a very large number of users, so that this software has a strong bearing on the socio-economic calculations, while custom built software will normally be in substantially less widespread use, that is to say the number of users will not have such a strong bearing on the calculations, whereas other factors play a more essential role.

The value of smaller software imports were not included in the socio-economic assessments of open-source and proprietary software. In the analysis, proprietary software is an imported product, while open source can be imported without payment of licence fees. The import value is estimated to be less than the total licence expenditure.

The working group has analysed this question in the light of the lack of statistical knowledge on IT operating expenditure broken down into specific types in the public sector and has come to the conclusion that desktop and servers probably constitute an ordinary administrative workstation, but that special software, both collective and split between workstations, cannot be assessed. Consequently neither can determine the proportion of the former products in total IT operating costs.

2.1.8 General conclusions from the presented surveys

The surveys presented in the previous paragraphs provided insights in fundamental features of FOSS. It shed a light on FOSS features, motivations, expectations and orientations. To sum up, the main factors that influence the selection between FOSS and proprietary SW solutions can be summarized in the following list:

- existence or not of a general governmental policy against FOSS
- flexibility
- interoperability

- costs of license fees
- training costs
- need for customisation
- lack of support
- integration factors
- functionality
- long term development issues
- maintenance issues
- customisation
- access to source code
- security issues
- code quality
- support for open standards and open development process
- vendor independence and flexibility
- user-friendliness
- software compatibility

2.2 Related Publications

2.2.1 ***Publications regarding technological factors that affect FOSS usage***

Taking in mind the analysis in [39], it must be argued that technological factors affect FOSS in a large scale. People that support the adoption of FOSS argue that shows more stable behavior than proprietary software. The authors of [14] claim that in organizations the use of OSS still has to be motivated on utilitarian grounds. Technological factors that show a relevance to OSS adoption include maturity, performance, stability, usability, security such as availability and quality of support.

Previous experience with FOSS plays a significant role in the ability to choose such kind of software. It is rather usual that 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 and support is available [39].

It is quite interesting to observe that several FOSS projects considered immature when measured with maturity models are mature enough for adoption, given that the adopting organisation has some OSS experience [56].

The same authors mention the maturity of the organization dealing with FOSS in [33]. 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 found that the maturity of the solution under review is dependent on its intended application within the organization.

Software maturity is a decision factor that is dependent on the environment in which the software is used [56], [31]. 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.

The key is to compare specific software products. Depending on the research, organisations perceive FOSS to be anything from more reliable [40] than alternatives to being inferior on all accounts. In Singaporean companies, OSS software is mainly used in systems infrastructure, resulting in increased stability and scalability [62].

It must be argued that an opposing view on FOSS maturity also exists. A common perception among organizations is that FOSS is an immature technology, not yet ready for commercial use. Many believe that goods available for free has to be of inferior quality when compared to paid proprietary software. Proprietary software vendors often use this perception in their marketing to plant uncertainty and doubt. Large scale adoption of FOSS and support of major software vendors however counters the perception of immaturity and several FOSS maturity models have been developed to help organisations determine the maturity of software [42].

Compatibility is a classic technology adoption factor relevant to FOSS adoption. FOSS adoption decisions are greatly influenced by the compatibility with existing technologies, skills and tasks [12], [40].Two types of incompatibilities exist when considering FOSS adoption, the first is incompatibility with existing legacy software, second is incompatibilities due to FOSS project forking [42].

In the case of long term adoption of FOSS, is more likely to change the adoption decision from applications only to enterprise wide infrastructure adoption of FOSS [31].

If a FOSS project forks, (in specific scientific domains like genetics), the user is stuck on one side of the fork. Software might become incompatible with the other side of the fork. If the other side of the fork is more successful, chances are that your side of the fork might soon become extinct and the incompatibility will endure until you switch [58]. From a pragmatic point of view the risk is reduced a pragmatic users typically only choose to adopt mature FOSS solutions with a very low risk of forking.

Security is a technology factor widely debated between proprietary software and FOSS proponents. Supporters of FOSS claim that many eyeballs also reduce the security risks associated with software. The likelihood of someone finding a security issue and fixing that issue is higher in the case of FOSS [58], [32]. Concerns also exist around intentional security holes. Access to source code ensures that organizations, and governments, can be confident that the software they are using is free of both intentional and non-intentional security holes [32].

Improved security is often cited in public sector adoption research as a key enabler of OSS adoption ([32], [48], [58]). It has also been found that improved security was a factor in Brazilian private sector adoption decisions ([48]). Collaboration enables the OSS development community to find and fix security issues much quicker [41].

Open standards play a role in both private and public sector adoption decisions. Open standards make integration possible in a heterogeneous computer environment [31]. Open standards also ensure the digital durability and future interoperability of information by storing it in a format that will be accessible in future [32].

Specifically, according to [31] moving from mainly technical issues in procurement to corporate IS governance presents FOSS with new challenges beyond outlining a business case for a particular FOSS application. They draw parallels to the business case for Commercially available Off-The-Shelf software products (COTS). Compared with COTS, FOSS products seem to have several advantages, but based on existing literature and a case study, they develop and discuss the hypothesis that a major barrier may be the “customer’s” uncertainty and unfamiliarity with FOSS vendor relationships. They find that corporate governance and architecture needs to be accounted for in both COTS and FOSS. This paper should be seen as a first step researching the fit between procurement and delivery models for FOSS. Some characteristics about FOSS are the following: software developed and maintained through the “Open Source model,” in which any developers contribute code to a common repository, software distributed as application programs, excluding e.g. code libraries, software maintained and developed by a mature, active organization, including and technological infrastructure (common software repository, website, mailing lists for users and developers).

The authors in [38] examine the impact of perceived benefits and drawbacks of FOSS on its adoption in 13 companies operating in the secondary software sector in Europe. The findings are analyzed using the adoption of innovation literature as a lens to reveal how technology, organizational, environmental and individual factors impact FOSS adoption.

Some technical benefits found are: Reliability (High availability and dependability of applications), Security (High security due to the availability of source code, the reduced threat of viruses and extra awareness of security in design phase of products), Quality (Enhanced quality from peer reviews and the quality of developers / testers), Performance (High performance in terms of capacity and speed), Flexibility of Use (Beneficial because it facilitates changes, customisation, experimentations and allows freedom of choice), Large Developer/Tester Base (Very beneficial as it ensures that OSS is quality software and is up-to-date), Compatibility (Great interest in conserving formats for better interoperability), and Harmonisation (Improved harmonisation in interoperability and practices/operations).

Respectively some technical drawbacks that were defined in [38] are: Compatibility Issues (Not significantly disadvantageous, but some compatibility problems with current technology, skills and tasks), Lack of Expertise (Employees lack OSS expertise - may be more about lack of awareness), Poor documentation (Documentation outdated or may have died in development), Proliferation of Interfaces (Results in confusion in deciding which one to choose Less Functionality Level of integration not as good as Microsoft) and Lack of Roadmaps with OSS Products (Makes it difficult for companies to see any strategic direction).

Weber in [59] has many insights into how an open source approach to almost any new idea or product development challenge can produce results better, faster, cheaper, and with much wider acceptance, than traditional top-down, command-and-control, closed-door management. As the author mentions, the idea that made Linux a worldwide phenomenon was the development of the General Public License (GPL). The GPL grants all users of Linux (or any other product it applies to) the freedom to use it for any purpose, to examine the source code, to redistribute it to others, and to improve it and share those improvements with others in the community. The author also answers one important question about open source: why would people give freely of their time and their intellectual ability to a product that will not directly generate income for them? Actually, there are plenty of reasons. For one, they get a better product to use for themselves or their companies. And of course they gain the personal satisfaction of proving they can add value.

Self-interest plays a big role here; and it isn't just for personal gratification. Many programmers have built their reputations within the software community on the basis of their high-quality contributions – and those reputations very often translate into well-paying jobs [59].

In the work [50] the authors explore online technical support of open source software by a study of postings to discussion boards. Their results indicate that there are several types of detail that are required by the help-givers to be able to diagnose and re-mediate help-seekers' difficulties. As a result help interactions may iterate somewhat inefficiently. These findings are compared with studies of telephone technical help lines for commercial software, and library reference interviews. By considering certain rather problematic interactions they can identify ways to improve the process.

Their pilot study revealed several interesting similarities and differences between open source technical help giving and both conventional technical help and library reference interviews [50] . As the authors mention, understanding how FOSS technical help is given is important, particularly as a way to improving the process. Unfortunately to date it seems to have been rather overlooked.

In [15] the author studies one of the main problems of OpenOffice which is that it does not support the usage of macros by default. As the author mentions, OpenOffice offers the possibility to each developer or to each simple user to create his own macro. The process involves the storage of a macro in a document library. In order to achieve that, the following steps are used: Creation of a library, creation of a module and then entrance of the macro. The user has the possibility to store a macro in the application library in a fully Integrated Development Environment. Moreover, openOffice offers the usage of breakpoints such as library management to each user. The above wok describes the way that libraries are stored and the categories that are divided (Application libraries & Document libraries). So the user after creating a macro can use the Macro Organizer, rename modules and libraries and add new ones. Therefore, the authors provide a way to overcome the "macro" problem in OpenOffice.

2.2.2 ***Publications regarding organizational factors that affect FOSS usage***

In [39], the author argues in the fact that several organizational factors that regard human behavior play a role in FOSS adoption decisions. It is important to notice that the skill levels in an organization will determine the amount of external support needed for successful adoption. The organizational success in adopting a new technology is also highly dependent on user acceptance and top management support.

An important factor that influences top management support of FOSS is the ability to find the right staff and competencies needed to adopt FOSS according to [40].

In rural areas such as South Africa, the authors in [33] found that 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.

An organizational factor that affects the adoption of FOSS is lack of awareness that can be remedied by having FOSS advocates and boundary spanners working in an organisation. Definitely boundary spanners are effective in connecting organisations to new technologies and provide the skills and knowledge needed for successful adoption [57] OSS champions successfully influence adoption decisions from within an organisation, reducing some of the individual uncertainty and fear [40]. The amount of influence FOSS champions have within an organisation is determined by the institutional limitations in the organisation and their position within the organisation [14].

One of the most important organizational barriers to technology adoption is resistance to change. Users that have second thoughts about adopting FOSS software are a significant barrier to FOSS adoption, especially in large scale migrations [31]. Users should be persuaded to use new technology without forcing them. Users are also mainly pragmatic and have little interest in the ideology behind the technology they are using [48].

Furthermore strong leadership and management support of FOSS is an important factor in FOSS adoption decisions and the FOSS migration process [48].

The availability of in-house OSS skills and knowledge are quite important adoption factors. A lack of skills leads to an increased dependence on external support ([48], [42]). Organizations with existing OSS skills are better equipped to mitigate risks associated with FOSS adoption and they also have lower training costs [24]. Lack of skills and awareness has also been identified as a barrier to adoption in South African [39] small and medium enterprises [17].

The lack of real world experience in FOSS adoption, could be regarded as an important organizational factor. If an organization adopts FOSS successfully, other companies follow. Evidence from real world OSS migration is required, typically in the form of case studies [40]. Evidence of successful migration could lead to organizations seeing the relevance to their own organizations. A perceived lack of relevance to operations to be a barrier to FOSS adoption was found in [24].

In the research [57] some results show that organizations with a strong background in IT were able to use FOSS without external support. Attempting to gain the most cost savings, some FOSS adopters choose to use FOSS without any commercial support. Dropping service-level-agreements to reduce costs increases the risk of failure and reduces the overall business value of using OSS as mentioned in [24].

2.2.3 ***Publications regarding cost factors that affect FOSS usage***

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 OSS adoption [40].

The business case for FOSS adoption is driven by lower costs, but 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 [31]. 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 [37].

It is interesting to observe that cost as a factor in FOSS adoption decisions depend on an objective measurement of cost. The authors in [48] found that for many companies, FOSS adoption is centered on value creation. The advantage however comes not only from costs which are saved but benefits from reliability, flexibility and a higher degree of innovation capability.

Developing countries, in general, adopt OSS due to cost advantages. The effect of software license fees are more pronounced in developing countries as it makes up a larger part of total system cost when taking into account hardware and software. Lower labour costs mean that license fees constitute a bigger percentage of IT costs [45].

One interesting example occurs in the German public sector where low cost is one of the main drivers of OSS 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, government uses OSS to save on license fees, keeping money that was previously paid to foreign vendors inside the country [48]. Through collaboration with local industries costs can be minimized and national competitiveness in software industries can be improved [32].

The importance of cost in OSS 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 ([31], [35]). In large scale adoption, proprietary software license discounts become a barrier to OSS adoption compared to proprietary software.

In the research [14] the authors investigate the cost aspect of FOSS. Lower costs are realized by two factors, the first is the absence of license fees for OSS software and software upgrades. The second cost factor is due to the fact that OSS often runs on commodity hardware and that it runs more effectively on that hardware [12]. A common misperception is that all FOSS is free; several FOSS vendors provide enterprise versions of OSS that come with certification and support services. Depending on their in-house skills, companies often need supported versions of OSS for successful adoption.

Software costs are also not only confined to license fees, the total cost of ownership (TCO) is used to determine the cost of adopting software and takes into account acquisition, operation, maintenance and disposal costs. The results of TCO studies are often contradictory and tend to be environment specific [56]. Objective cost measures like TCO and return on investment (ROI) are often not used in FOSS adoption decisions, despite the realization that cost should play a significant role in adoption decisions [62]. Reduction in TCO is stated as one of the main reasons for FOSS adoption in Brazilian companies as it provides the ability to employ low cost platforms and reduce security, maintenance and repair costs [48].

3 Factors Affecting FOSS Usage

In the paragraphs that follow, we present the main technical, social and organisational factors that affect the FOSS usage and should be taken into account during the software selection procedure.

3.1 Technical factors

3.1.1 ***Functionality***

It refers to the degree that a program integrates and is compatible with existing components. It also means if there are relevant standards and if the program supports them and what hardware, operating systems and related programs are required [60].

3.1.2 ***Support***

The term "support" covers several areas:

- training users on how to use the product,
- installing the product, and answering users who have specific problems 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:

1. they can choose a traditional commercial support model, where they pay someone (typically a company) to provide support,
2. they can choose to provide support in-house (designating some person or group to do the support),
3. they can depend on the development and user community for support (e.g., through mailing lists).

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.

In many cases, large companies will choose to use a more traditional support model - that is, they'll pay some company to give them all that support. Unlike proprietary support (which is usually only provided by the proprietary vendor), there may be several competing companies offering support [60].

3.1.3 ***Maintenance / Management / Longevity***

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's far more likely that the program will be useful in the future. FOSS maintenance options are essentially the same as those for support, and in reality maintenance and support are not completely separate [60].

Software maintenance and management are time and resource consuming processes. Fault detection and correction are the main activities in the software maintenance and management. It is preferred that these activities are performed and finished in the software development process, i.e. before the software release. Therefore, identifying parts of the software where testing efforts should be focused can help software engineers and project managers, in testing, inspections, and restructuring efforts towards these critical parts of the software. As a result, developers can use their resources more efficiently to deliver higher quality products in a timely manner because applying equal testing and verification effort to all parts of a software system has become cost-prohibitive. It is therefore important to invest in fine-grained comparison and versioning tools to carefully track changes and compare new versions of frameworks to determine the impact of upgrading to a future release [16].

3.1.4 ***Compatibility***

Software/hardware compatibility failures often had to be solved during the implementation of a FOSS project. This is a critical aspect, particularly for large scale migration projects in which a compatibility failure could threaten the entire project [60]. To avoid this, possible failures should be foreseen prior to implementation and specific issues should be addresses such as:

1. availability of hardware drivers,
2. compatibility of hardware units with operating systems,
3. collaboration of open source systems and applications with existing proprietary software systems.

3.1.5 ***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 [60]. 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.

3.1.6 ***Security***

Evaluating a product's security is complicated, in part because different uses and different environments often impose different security requirements on the same type of product. One step toward solving this problem is the identification of security requirements [60].

Independent evaluations of the software can give some valuable information. There are many tools that carry out this procedure. It's important to be aware of the limitations of such tools; if a typical security code scanner reports "no security bugs", that simply means the tool didn't find any, not that there are truly no security vulnerabilities. Many security vulnerabilities stem from certain typical mistakes that are detectable by these tools, so eliminating them can greatly increase the security of the program.

3.1.7 ***Scalability***

Scalability suggests the size of data or problem the program can handle. It is a very crucial factor for selecting the appropriate software and for example, if someone expects the program to be able to handle unusually large datasets, or be able to execute on massively parallel or distributed computers, before selecting the appropriate software there should be some evidence that the program has been used that way before [60].

3.1.8 ***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 not present a full picture. Project mailing lists may include more detailed performance information [60].

3.1.9 ***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 at all. 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, to see how easy it is to use.

For applications intended for direct use by users, there are basically two kinds of human-machine interfaces used by most of today's programs: a command-line interface and a graphical user interface (GUI). These kinds are not mutually exclusive; many programs have both. Command line interfaces are easier to control using programs (e.g., using scripts), so many programmers and system administrators prefer applications that have a command line interface available. So, if the application will need to be controlled by programs sometimes, it is a significant advantage if it has a command line interface. There are alternative user interfaces for special purposes [60].

3.1.10 ***Flexibility / Customizability***

Flexibility and customizability are two highly interrelated attributes.

- **Flexibility**: measures how well a program can be used to handle unusual circumstances that it wasn't originally designed for.
- **Customizability**: measures how well someone can customize the product to fit into his/her specific environment.

FOSS programs have a significant advantage over most proprietary programs in both flexibility and customizability: any FOSS programs can be modified as much as necessary for your circumstance. However, taking advantage of this may require either programming skill or paying someone with such skills to do so. Also, some FOSS programs are easier to extend than others. Moreover, a user could look to see if there mechanisms that make the program easier to make it fit for his/her specific purposes, such as templates, "plug-ins", a programmer's application programming interface (API), or a command language [60].

3.1.11 ***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 -- that way the best product can be chosen (instead of being locked into one vendor's product), and change later. FOSS products typically implement relevant standards, simply because there's usually no good economic reason not to [60].

Therefore, one of the motivations for the use of FOSS is to try to achieve vendor-independence, which is to retain the ability to smoothly change software products or producers in future [18], [30]. However, this can conflict with with the requirement that the new software must be integrable with already installed operational software.

”Buyers who give priority to the latter criterion instead of using a general requirement for open standards or vendor independent inter-operability remain locked in to software they previously purchased” [16].

3.1.12 ***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. Trailability of an innovation is hypothesized to be positively related to the adoption of that innovation. With respect to open source software, it can be argued that open source software is easier to try out than commercial software, because a full version of the software can be freely downloaded from the Internet [55]. It is very important to be able to try a software before using it in a production environment. Although the trialability of open source software is not questioned, a wide range of opinions exists on whether open source software is easier to try out than commercial software. Some organizations consider open source software easier to try out, because it can simply be downloaded from the Internet, without cost and without any administration. However some others do not distinguish between the trialability of commercial and open source software, because it is possible to obtain demo or trial versions of commercial software. They admit however that using these trial versions may be a bit more cumbersome since most vendors require prior registration In every case, the trialability of open source software is supposed to be an important advantage [12], [13].

3.1.13 ***Data Migration***

Data is stored and managed by database applications. Virtually all public administrations have huge databases. Often this data is of critical importance and huge (financial) resources have been and are allocated to collect, organize, and maintain the data. It is important to divide the data into categories namely [16]:

1. Data which can be discarded.
2. Data which is useful and in open format such as PDF or Postscript, or can be easily translated into open format. The cost should be considered.
3. Data which must be kept but which is in a legacy closed format which cannot be easily translated into an open one. This data may need copies of the legacy software.

3.1.14 Technical issues for the migration to FOSS solutions

In this paragraph we present some of the problems related to the migration to FOSS solutions may be the following [54]:

- 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

Co-ordination standards and mechanisms

Open standards have, without a doubt, been a strong contributing factor to a minimising of direct lock-in effects and a high degree of competition within many areas. GSM is a well-chosen example of how consumers as well as suppliers and vendors in the area of mobile telephony have been profited from standardisation. The opportunities for standardisation differ, however, between different areas of technology, due to prevailing market strength of a player and competitive vendor/customer situations [54].

When choosing the proposals and bids of various vendors it is of course suitable that special attention be given to assessing how applicable standards are met. Here one should be especially attentive to the risk of proprietary additions to open standards. A vendor can often say that it supports a standard completely, but append a number of enhancements which lead to lock-in effects if they are used, The value of these enhancements must be weighed against the costs of lock-in. FOSS minimises, of course, the possibilities for a vendor to distort standards in this way, since all other vendors automatically have access to the enhancements [54].

Foss Combination with proprietary software

Combining FOSS with proprietary software is possible, depending on the manner of “combination” and on the specific license of the software. Of all the common FOSS licenses, the GNU GPL license is the one that requires the most care. It defines “combination” as follows: Mere aggregation of two programs means putting them side by side on the same CD-ROM or hard disk. This term is used where they are separate programs, not parts of a single program. In this case, if one of the programs is covered by the GPL, it has no effect on the other program. Combining two modules means connecting them together so that they form a single larger program. If either part is covered by the GPL, the whole combination must also be released under the GPL—if you can't, or won't, do that, you may not combine them [21].

In this case, if one uses a proprietary application in a FOSS operating system environment, the proprietary application is unaffected by the licensing of the FOSS system. An example of this is running an Oracle database on a GNU/Linux operating system.

An example of combining programs would be writing a GUI application using the Gnome application framework. The Gnome application framework speeds up the development of any GUI program by supplying functionality developers who would otherwise have to write from scratch. Gnome is licensed under the GPL. Because the completed application program (after a compiler has been through it) would contain source code from the Gnome application framework, the entire application would have to be licensed under the GPL [21].

3.1.15 *Other technical factors*

According to [39] some other technical factors that affect FOSS usage are the following:

- Previous experience with FOSS
- Software maturity that depends on the environment in which the software is used
- Open standards
- Quality issues
- Large Developer / Tester Base
- Harmonisation
- Lack of Expertise
- Poor documentation
- Proliferation of Interfaces
- Lack of Roadmaps with FOSS Products

3.2 Social factors

"Localization involves taking a product and making it linguistically and culturally appropriate to the target locale (country/region and language) where it will be used and sold."

Community identification, self satisfaction, and fulfilment that arise from writing programs are considered as the motivators of FOSS developers, since their desire is to fulfil their personal needs, which was the case in both the PERL and Apache projects [28].

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 from the FOSS survey shows that the major reasons of developers' participation in FOSS software development are:

- 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 [36] is a key motivator that can be used in technology transfer.

3.2.1 Localization Industry Standards Association

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 minimally localized version of any FOSS. While the construction of a completely localized software platform is no small feat, it is at least possible.

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 in order to be used 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 are also an increasing number of projects producing customised software for public administrations [61].

3.2.2 Economic Factors

Although the low price of FOSS products is the primary factor for using these products, this section introduces other economic perspectives, not only in using FOSS but also in developing products. Four economic incentives have been identified in [49] for the adoption of FOSS software and support its development by governments

- 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

Although most of the developers (46%) do not earn money from FOSS developments, developers do anticipate direct or indirect monetary rewards. Direct rewards for individuals are identified as the revenues from related products and services such as commercial consulting, training, distribution, support and implementation services, or rewards from current or future employers to seek higher wages or attractive job positions or career benefits [11] , [1].

3.2.3 ***Local Policies***

Some governments and other organizations have specific policies preferring FOSS programs over proprietary programs. This is rare; what's 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 [60].

3.3 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.

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 [8].

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 argue that a centralized structure improved migration to FOSS [8].

The cities' experiences with migration to FOSS also demonstrate a more complex relationship between organizational and technological change than what appears in scholarly literature. While technological change is often viewed as the product of organizational characteristics, the three case studies point to an inverse relationship: New technology changes the organization [8].

Respondents in each city explained that migrating to FOSS led to virtual and physical organizational changes. Virtual organizational change refers to how the cities managed their computer software systems. The policy to migrate to FOSS forced each city to take stock of its IT hardware and software because without such an assessment it wouldn't have been possible to implement the migration policy. In some cases, cities conducted the assessments on their own. In other cases, cities relied on assistance from private-sector partners. As cities addressed their virtual organizations, several also made changes to their physical organizations. Cities took stock of their IT staffs, identified redundancies and moved toward a more centralized IT support structure [8].

Finally, respondents in the three cities reported that the switch to FOSS improved their internal capacity and increased employees' willingness to innovate. Because the benefits from FOSS derive from working with computer code, the advantages of open source increase as the IT staff's expertise increase. While Schwäbisch Hall and Munich relied on contractors to aid the implementation process, all three cities were and are committed to doing as much of the IT work in-house as possible. And as the skill level of the IT professionals increased, so did the motivation to innovated. Each city reported developing new programs and applications, which were shared with other cities, as well as the broader open source community. Schwäbisch Hall, for example, recently developed a new council information application to provide materials and minutes to city parliamentarians. Munich and Treuchtlingen also have developed dozens of new applications. And Munich was recognized with the European E-Learning Award in 2007 for the learning platform the city developed to teach staff how to use open source software.

In sum, the German cities' experiences suggest that the decision to switch from proprietary to open source software is neither easy nor obvious. It depends on a range of factors — administrative capacity, political backing and organizational structure. Yet, the experiences of the three cities underscores that while FOSS may not be appropriate for every circumstance, it should at least be considered by local officials as a viable, perhaps even superior, alternative to its proprietary counterpart [8].

3.3.1 ***Legal / License issues***

Legal issues are another important attribute, and they are primarily defined by a program's license. Thus, an organisation should examine the license requirements for each considered program as well as their implications in the current country.

Unlike most of the other factors, this factor is sometimes overlooked when evaluating proprietary software, and that's a mistake.

When someone is evaluating proprietary software, he must be sure to examine its licensing terms such as its End User License Agreement (EULA). Some EULAs have clauses that may be found unacceptable, such as allowing a vendor to gain access to an organization's computers and networks to do compliance audits, obligating you to large fines if the vendor finds unlicensed copies (even if the copies were not sanctioned by the organization), allowing the vendor to remotely disable the software without a court decision or other legal protection, forbidding the disclosure of evaluations (such as benchmarks) to others, limiting transfer or use of the program (such as limits on data volume), or allowing the proprietary program to send private information to the vendor. Even if others find the EULA conditions acceptable, the interested user may not find the EULA conditions acceptable for the organization [60].

3.3.2 ***Other organizational factors***

According to [39] some other organisational 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

3.4 Benefits using FOSS

The adoption of FOSS concepts in developing countries promotes local research and development, rather than external suppliers or importing technological products. Also, FOSS can provide the leverage for locally developed skills, increase local talents participation, minimize investment risks, and increase cost saving [1].

The wide use of FOSS increases the utility of the technology with the increase in the network size. This concept is known as the network effect where users provide feedback and standardize the use of the technology which in turn is evident for the usefulness of the technology [51]. From the industry and business point of view, FOSS is a boost in the establishment of start - up firms, offering new business models for existing products. Such activities mean support or maintenance contracts, alliances to establish standards, or different licenses for customized models of FOSS technologies.

One more advantage of FOSS is that the FOSS content is courseware that can improve knowledge accessibility and education. It would also improve the teaching and learning approaches, and curriculum through peer review which in turn lowers the cost of course development.

One more considerable advantage of FOSS is the ability to easily modify or change a product for a certain group of users, i.e. to make it simple and functional. One can create a specific development environment, an application for e-services for the general public or a customised desktop or workplace computer with modified functionality. Products become more operatively secure and easier to administrate and maintain [1].

Some more advantages of FOSS are listed below:

- higher stability
- high level of security
- none or low licensing fees
- possibility to modify source code
- ample access to IT specialists
- independence from major software vendors

In the following paragraphs several benefits of FOSS are discussed.

3.4.1 ***Security***

While there is no perfectly secure operating system or platform, factors such as development method, program architecture and target market can greatly affect the security of a system and consequently make it easier or more difficult to breach [61].

Three reasons are often cited for FOSS' s better security record:

- **Availability of source code:** The availability of the source code for FOSS systems has made it easier for developers and users to discover and fix vulnerabilities, often before a flaw can be exploited. Many of the vulnerabilities of FOSS were errors discovered during periodic audits and fixed without any known exploits. FOSS systems normally employ proactive rather than reactive audits.

- **Security focus, instead of user-friendliness:** FOSS can be said to run a large part of the Internet and is therefore more focused on robustness and functionality, rather than ease of use. Before features are added to any major FOSS application [26], its security considerations are considered and the feature is added only if it is determined not to compromise system security.

- **Roots:** FOSS systems are mostly based on the multi-user, network-ready Unix model. Because of this, they come with a strong security and permission structure. Such models were critical when multiple users shared a single powerful server—that is, if security was weak, a single user could crash the server, steal private data from other users or deprive other users of computing resources. Consequently, vulnerabilities in most applications result in only a limited security breach.

3.4.2 Reliability / Stability

FOSS systems are well known for their stability and reliability. There are many anecdotal stories of FOSS servers functioning for years without requiring maintenance [25]. However, quantitative studies are more difficult to come by. Here are two of the studies conducted to date:

- In 1999 Zdnet ran a 10-month reliability test between Red Hat Linux, Caldera Systems OpenLinux and Microsoft's Windows NT Server 4.0 with Service Pack 3. All three ran on identical hardware systems and performed printing, web serving and file serving functions. The result was that NT crashed once every six weeks but none of the FOSS systems crashed at all during the entire 10 months [7].

- A stress test using random testing stressed seven commercial systems and the GNU/Linux system in 1995. Random characters were fed to these systems, to simulate garbage from bad data or users. The result was that the commercial systems had an average failure rate of 23 percent while Linux as a whole failed nine percent of the time. GNU utilities (software produced by the FSF under the GNU project) failed only six percent of the time. A follow-up study years later found that the flaws identified by the study were all fixed in the FOSS system, but were generally untouched in proprietary software[7].

3.4.3 Open standards and vendor independence

Open standards give users, whether individuals or governments, flexibility and the freedom to change between different software packages, platforms and vendors [2]. Proprietary, secret standards lock users into using software only from one vendor and leave them at the mercy of the vendor at a later stage, when all their data is in the vendor's proprietary format and the costs of converting them to an open standard is prohibitive.

The authors of the paper "Free / Libre and Open Source Software: Survey and Study" produced by the International Institute of Infonomics in the Netherlands [23] also argue against use of proprietary software in government. They say that one major argument against the implementation of proprietary software in the public sector is the subsequent dependency on proprietary software vendors. Whenever the proprietary standards are established the necessity to follow them is given. Even in an open tender acquisition system, this requirement for compatibility with proprietary standards makes the system biased towards specific software vendors, perpetuating a dependency.

Another advantage of FOSS is that they almost always use open standards. This is due to two primary reasons:

- **Availability of the source code:** With the source code, it is always possible to reverse-engineer and document the standard used by an application. All possible variations are plainly visible in the source code, making hiding a proprietary standard in FOSS systems impossible. Proprietary software, however, are much harder to reverse-engineer and in some cases are deliberately obfuscated.

- **Active standards compliance:** When established standards exist, such as HyperText Markup Language (HTML), which controls how web pages are displayed, FOSS projects actively work to follow the standards faithfully. The Mozilla web browser, a FOSS effort, is fully compliant with many standards from the World Wide Web Consortium (W3C). Webstandards.org notes that Mozilla is one of the most compliant browsers available today [27]. Compliance with standards is due to the FOSS development culture, where sharing and working together with other applications are the norm. It is also much easier to work with a globally dispersed group of developers when there is a published standard to adhere to. Using FOSS systems as a means of gaining vendor independence has been raised in several areas. A report to the UK Government concludes that “the existence of an OSS reference implementation of a data standard has often accelerated the adoption of such standards, and recommends that the Government consider selective sponsorship of FOSS reference implementations.

3.4.4 ***Reduced reliance on imports***

A major incentive to adopt FOSS systems is the enormous cost of proprietary software licenses. Because virtually all proprietary software in developing countries is imported, their purchase consumes precious hard currency and foreign reserves. These reserves could be better spent on other development goals.

The European study in [23] also notes that, “The costs of this more service-oriented model of open source are then also normally spent within the economy of the governmental organization, and not necessary to large multinational companies. This has a positive feedback regarding employment, local investment base, tax revenue, etc.”

3.4.5 ***Development of local software capacity***

It has been noted that there is a positive correlation between the growth of a FOSS developer base and the innovative capacities (software) of an economy. The report from [23] lists three reasons for this:

- Low barriers to entry: FOSS, which encourages free modification and redistribution, is easy to obtain, use and learn from. Proprietary software tends to be much more restrictive [3], not just in the limited availability of source code, but due to licensing, patent and copyright limitations. FOSS allows developers to build on existing knowledge and pre-built components, much like basic research.

- FOSS as an excellent training system: The open and collaborative nature of FOSS allows a student to examine and experiment with software concepts at virtually no direct cost to society. Likewise, a student can tap into the global collaborative FOSS development network that includes massive archives of technical information and interactive discussion tools.

- FOSS as a source of standards: FOSS often becomes a de facto standard by virtue of its dominance in a particular sector of an industry. By being involved in setting the standards in a particular FOSS application, a region can ensure that the standard produced takes into account regional needs and cultural considerations. The FOSS developmental approach greatly facilitates not only innovation but also its dissemination.

3.4.6 Piracy

Software piracy is a problem in almost every country around the world. The Business Software Alliance estimates that software piracy in 2002 alone cost US$13.08 billion. Even in developed nations where software is affordable in theory, piracy rates were as high as 24 percent in the United States and 35 percent in Europe. Piracy rates in developing countries, where lower incomes make software far more expensive, are upwards of 90 percent [29]. Software piracy and lax laws against it can and does hurt a country in many ways. A country with poor protection for Intellectual Property Rights (IPR) is not as attractive to foreign investors. Membership in the World Trade Organization (WTO) and access to its benefits are strongly affected by the level of protection given to IPR in a country. Finally, a culture of software piracy hurts local software development, as there is less incentive for local software developers to create a local product.

3.4.7 Motivations for firms‘ Open Source activities

Most FOSS activities by the major software makers are in the field of Linux. Even those companies that do not visibly contribute to FOSS development are in many cases at least passive Linux supporters by having ported some of their software to the Open Source operating system [46].

As firms typically have the target to make profits and as they cannot earn income directly from selling the Open Source software they produce, the justification for the OSS engagements must come in some way from complementary goods or other indirect effects. The economics literature points out especially the strategy to sell complementary products. RedHat, SuSE and the other Linux distributors are good examples for companies providing additional products and services related to Linux. Selling additional hardware, as IBM does, is another example [20].

Four major motivations behind the companies‘ Open Source activities have been identified. These motivations are [46]:

- Standardisation: overcoming the ghost of Unix wars
- Open Source software as low-cost component
- Strategic considerations
- Enabling compatibility

3.4.8 *Motivations for developing Open Source / Free Software*

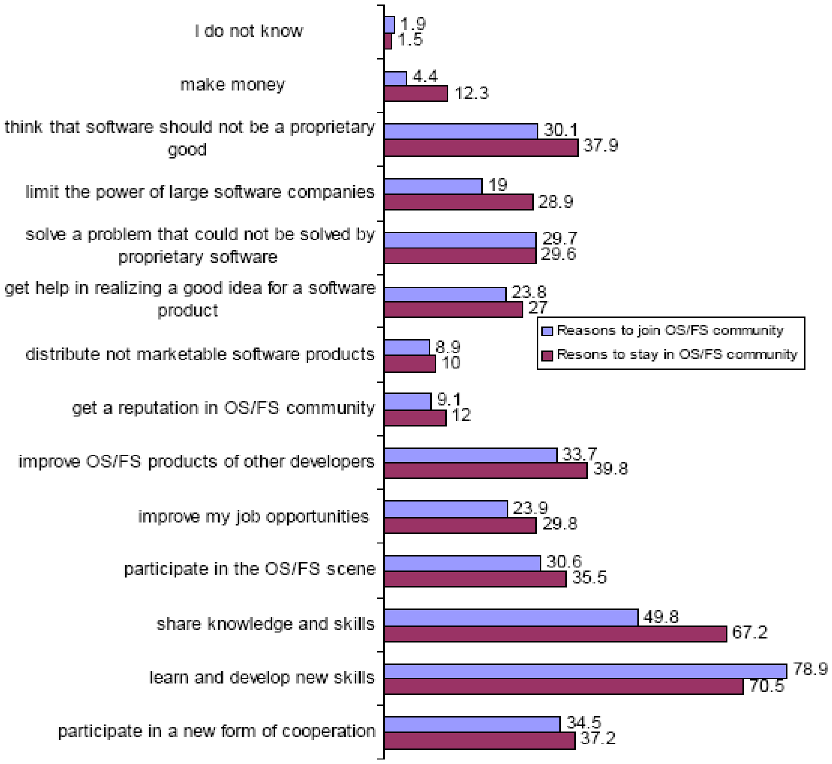


Figure 35: Reasons to Join and to Stay in OS/FS Community

Many researchers have been interested into the motives of people to join the Open Source / Free Software (OS/FS) community from two different perspectives: which motives have been causal to join the community and which motives keep the developers staying in this community .

Figure 35 illustrates the answers two these two questions. Most of the respondents ticked reasons that resided on the individual skills level, but there is also evidence of a social aspect [46].

3.4.9 *Incentives for Using FOSS*

Although low cost is the most obvious factor for the adoption of FOSS products, the transaction costs of licensing and acquisitions negotiation can be reduced [46]. This stems from the fact that the information is available and licensing is simple. Some of the reasons that support the use of FOS products in firms follow

- To attain direct involvement in defining a software's features or adding them to increase the product's usability
- To acquire direct technical support from the developers
- To reduce that training and deployment costs by accessing on-line forums, mailing lists, and documentation

3.4.10 *Summary of Positive effects*

There are many factors that can lead Open Source to success. Here we summarize the main factors:

- simpler license management
- reduced dependence on a product, less risk for ”lock-in” effects
- lower costs overall
- increased competition
- increased quality and stability
- increased activity on part of local/domestic businesses
- increased security
- open formats simplify communication with general public
- decrease power of monopoly/oligopoly of commercial companies

3.5 Inhibitors/barriers

For all the benefits FOSS brings, it is not suitable for every situation. There are areas where FOSS needs improvement.

3.5.1 *Administrative barriers*

Lack of business applications

While there are many FOSS projects out there today, there are still many areas that lack a full-featured product, especially in the business world. The porting of Enterprise Resource Planning platforms such as SAP and Peoplesoft have helped cover the high-end application market, but the Small and Medium Enterprise (SME) market is still poorly served. Basic, polished accounting applications such as Quickbooks, Peachtree or Great Plains do not have FOSS equivalents at this time. This problem has come about in part due to the scarcity of people competent in both technical and business subjects. Technical developers who encountered problems and wrote software to “scratch an itch” started most of the existing FOSS projects today. These projects are usually fairly technical in nature, such as the creation of web servers, programming languages/environments and networking tools. It is rare for a software developer to encounter accounting problems, for example, and have the business knowledge to create a technical solution [61].

3.5.2 *License policy*

FOSS is released under a variety of different licenses. There are two primary types of licenses and countless variants. The two main licenses are the GNU [52] [53] (recursive acronym for GNU’s not Unix) General Public License and the BSDstyle licenses [6]. A more detailed listing of licenses can be found on the FSF’s website at http://www.fsf.org/licenses/license-list.html.

3.5.3 *Functionality (interoperability, compatibility)*

Established FOSS lacks the extensive documentation and user-friendliness found in commercial software [34]. The primary focus of early FOSS developers was functionality. Creating a program that worked well was far more important than ease of use. Besides the dearth of high-quality documentation, there are also user interface issues with FOSS Graphical User Interfaces (GUI). Because the GUI element in most FOSS systems is not a single element but a collection of different projects glued together, the behaviour of the GUI elements differ greatly. Command-to-save data differ from one program to another, quite unlike proprietary desktop operating systems. Cutting and pasting between different programs can be wildly inconsistent or even impossible. While there is significant ongoing work to unify the desktop, the desktop is likely to remain inconsistent for some time to come.

FOSS systems, especially on the desktop, are not completely compatible with proprietary systems. For organizations that have already invested massive amounts of capital into proprietary applications and data storage formats, attempting to integrate FOSS solutions can prove to be prohibitively expensive. Changing proprietary standards, which is often aimed at preventing the integration of alternate solutions, exacerbates this problem. In time, as organizations shift from proprietary to open standards, this problem should be reduced [61].

3.5.4 *Issues related to intellectual property rights*

Economists have long since argued that society would invest insufficient resources in basic research in all market regimes. While monopolists do have a lower incentive to innovate than firms in perfect competition, the latter have the problem that a competitor can quickly exploit the useful new knowledge when it is unprotected. The obvious solution to this problem is IPR protection, e.g. in the form of patents. However, these introduce another inefficiency. Since the knowledge has been produced and can be distributed almost without cost, it is inefficient not to do so. This is a dilemma without a simple solution.

As firms are voluntarily choosing to participate in Open Source projects, the observed volume of Open Source projects by firms can be interpreted as the (local equilibrium) outcome of their research investment. As firms can protect most of their intellectual property in the domain of software development, it can be assumed that they only give as much intellectual property away in the form of Open Source software as is optimal for them.

The question to address is thus, whether this level of Open Source activity could be increased without weakening the rate of innovation within these companies. Are there certain peculiarities of the Open Source process that keep such firms from participating that under different regimes would be willing to make their knowledge available to others?

One issue pointed out, for example by Microsoft, is the viral nature of the GPL (which governs Linux) and especially ambiguities in its virality, which supposedly makes it difficult to build commercial software on top of Open Source software. It has to be taken in account that unclear legal implications might indeed be issues keeping companies from taking part in those Open Source projects governed by such licenses or from including such software as infrastructure components into their products.

However, these ambiguities do not concern the release of formerly proprietary software as Open Source, as companies are free to choose the license they want when doing so. The strictness of the GPL is one reason why many software companies use BSD Unix, governed by the more liberal BSD license, as foundation for their commercial software [61].

3.5.5 *Issues for public support of Open-source activities*

Public support of Open Source activities requires the following three issues to be taken into account. First of all, the support should be structured in a way that lets the market decide about which projects are useful. This is, e.g., the case when supporting infrastructure for Open Source development. Secondly, support should go to those kinds of Open Source projects that provide software closest to basic research, i.e. Infrastructure- like software that can be used as component in many other kinds of software. And thirdly, the license regime of the supported Open Source software projects should be such that the results can be used in as many ways as possible. This would exclude strong viral license regimes such as the GPL [61].

3.5.6 *Internal cultural issues*

Technical obstacles are perceived to be less a major barrier to implementation of open source than internal cultural issues within councils themselves. A widespread perception exists in local government that open source software is too risky for councils to consider. If adoption of open source software in local government is to be increased, there is a pressing need to raise councils' confidence in taking the open source approach. Exemplar authorities who can demonstrate successful implementation to councils, along with greater endorsement of open source from central government, both have a powerful role to play [61].

3.5.7 *Interoperability, proprietary standards and vendor lock-in*

Interoperability is for most institutions the main reason not to use open source software. Since the standards of proprietary software are normally not open, it is hard for competitors - be they for profit or non-profit, proprietary or open source - to ensure that their software is able to process data produced by proprietary software (e.g. graphs or tables in word processors). By their dominant market position, proprietary software vendors can thereby enforce a kind of de facto standard, e.g. on office software, which then - despite and because of the fact of being closed - enhances the vendors' market position. This is of course a self enforcing process. Consequently one major argument against the implementation of proprietary software in the public sector is the subsequent dependency on proprietary software vendors [4].

Whenever the proprietary standards are established the necessity to 'follow' them is given. Even in an open tender acquisition system, this requirement for compatibility with proprietary standards makes the system biased towards specific software vendors, perpetuating a dependency. This basically is due to two reasons: First of all software owners have to upgrade the software, even if there is no internal reason or interest in doing so. Otherwise they risk facing a situation where their programs are not capable to process documents and files, created by newer versions of the same product. The second coercion to upgrade evolving from this dependant situation is the ending support of 'older' versions.

This situation has thus major consequences for the cost side of IT management. Additionally for the costs for new licenses and update implementation, software users constantly have to be trained in new program versions. User performance in the phase after the implementation of the software always decreases. On the hardware side this dependency leads to an increase of expenditure. Newer proprietary software, normally, requires better hardware performance. Not enough RAM memory or processor speed for instance then very often leads to new, unnecessary investments in that area. Therefore the lifetime for hardware is much shorter.

This has often been described as the typical lock-in situation: The system is working with proprietary standards and is as such in itself interoperable. Migration to another reliable and interoperable technology is requires much effort and a high cost. The longer the situation goes on, the worse it becomes. After a while the software vendor does not have to fear competition, since the client has to take its product anyway. A typical - at least de facto – monopoly situation evolves in which the vendor dictates prices, conditions, and quality. Consequently liberation from this situation is advantageous for the buyer [61].

3.5.8 ***Security***

Regarding the question of data security, open source software is believed to be less vulnerable than proprietary software due to a simple reason: the source code is available. Proprietary software hides the code. For administrators proprietary software is a "black box" they have to trust regarding its security. Not only intentionally created "backdoors", but also conventional bugs are not perceivable. For instance there are much more defacements of websites running on proprietary software than on open source software. Open source software developers actively ask to check security gaps. If there is one, awareness of this security problem, and possible remedies, become public immediately.

Objections of proprietary software vendors that no open source software developer guarantees the security of the product are valid. However the license conditions of proprietary software generally excludes any liability resulting from damages arising from security gaps within the software. Normally just a substitution of the storage medium (e.g. the hard drive) is provided in case of harm causing defects of software. This scenario is hardly reported and results in any case only to comparatively low costs. The real damage, such as the loss of data, wrongly executed commands, or the loss of possible profits is not compensated. Producers or vendors of proprietary software do in general not give guarantees for the correct functioning of the programs. Indeed, most End-User Licence Agreements (EULAs) for proprietary software explicitly exclude any liability arising from security or other "bugs" in the software product [20].

In addition to the not excluded possibility of an open inspection of the source code by the scientific and developers' community, proprietary software producers in many cases include non-disclosure clauses in the license agreements. These contractual regulations prohibit the software owner from publicly revealing discovered bugs within the software. This non - communication situation then leads to a much less transparent and thereby to a much less secure condition under which the software is used. In general, this issue of "security versus obscurity" has been widely discussed by the academic and professional security and cryptography communities, with the universal conclusion that true security never arises from obscurity (i.e. the hiding of internal structures, such as source code) [20].

3.5.9 ***Summary of negative effects regarding FOSS***

In brief, some of the negative effects regarding FOSS are the following:

- lack network of companies/communities for support FOSS applications
- absence of national/EU policy for using FOSS in public sector
- lack of printed publications about FOSS (in nature language) for simple users
- absence of national/EU education policy (closely linked with national policy for using FOSS) for FOSS in public education system (primary/secondary schools, Universities etc.)
- fragmented development of FOSS (particular solution depends on quality of particular developer, some parts of one FOSS application are good, some are worse or bad)
- unpredictable development of FOSS applications (eg. OpenOffice - core of developers leave to build Libre Office)
- user friendly solution - lot of applications in FOSS are only for ICT people or FOSS freaks, not for common users
- psychological resistance among decision makers

4 Guidelines for selecting between FOSS and proprietary SW solutions

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

Most FOSS programs have a great deal of publicly available information that isn't 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's working (or not), and so on.

An even more fundamental difference between FOSS and proprietary programs is that FOSS programs can be changed and redistributed by customers. This difference affects many factors, 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 that case, the user still needs to be able to evaluate FOSS programs, because he/she will always need to know how well a given program meets his/her needs, and there are often competing FOSS programs .

4.1 Form 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 [60].

4.2 Identification of potential software solutions

A combination of techniques should be used in order to make sure that something important is not missed. An obvious way for the interested user is to make a questionnaire, if other users (organisations 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 at 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 a terrible mistake to not consider them. It is advised to the interested user to ask 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 proper to make a more 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's 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's 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 Defining 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, 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

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, transition costs such as data transition, etc.)
- Market Share

- Support
- Trialability
- Maintenance / Longevity
- Compatibility
- Reliability / Availability
- Security
- Scalability
- Performance
- Usability
- Flexibility / Customizability
- Interoperability
- Legal / License issues
- Local Policies
- Possible Data Migration
- Organisational factors
- Human factors (e.g. resistance of personnel to change SW, maturity of the personnel, etc.)
- Other environmental and social factors

The benefits, drawbacks, and risks of using a program can be determined from examining these attributes. The attributes are the same as with proprietary software, of course, but the way that a user should evaluate them with FOSS and 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.

4.6 Perform an analysis of the top selected software solutions

After the evaluation, the organization picks the top candidates, and performs a more analysis of them. This step is, for the most part, done 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 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.

REFERENCES

[1] Alkhabit J., Anis M., and Noori H., “Open Source: The next big thing in technology transfer to developing nations, International Association for Management of Technology, IAMOT 2008 Proceedings.

[2] Aziz H., Gao J., Maropoulos P., Cheung M. “Open standard, open source and peer-to-peer tools and methods for collaborative product development”, Elsevier, Computers in Industry, Volume 56, Issue 3, Pages 260-271,April 2005.

[3] Babcock C. “F/OSS code: a corporate building block”, Interactive Week, 2001.

[4] Behlendorf B. “F/OSS as a Business Strategy”, in In DiBona, C., Ockman, S., Stone, M. Eds. F/OSSs: Voices from the F/OSS Revolution. O'Reilly & Associates, Sebastopol, CA, 1999.

[5] Robert Francis Group, Total Cost of Ownership for Linux in the Enterprise. July 2002.

[6] Bobko P. “Open-Source Software and the Demise of Copyright”, Rutgers Computer & Tech. LJ, 2001.

[7] BOSS, “Facts About FOSS”, Bharat Operating System Solutions, http://bosslinux.in/Resources

[8] CIO-Zone, CIO-SUMMIT, “5 Factors for Open Source Success”, Published by govtech.com.

[9] 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.

[10] 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.

[11] Dalle J., Jullien N. “NT vs. Linux, or some explorations into the economics of free software,” In G. Ballot and G. Weisbuch, eds, Application of simulation to social sciences, Paris, France: Hermès, 399-416, 2000.

[12] Dedrick J, West J (2004) An exploratory study into open source platform adoption. In: Proceedings of the 37th Hawaii International Conference on System Sciences.IEEE Computer Society, Washington, DC.

[13] Dedrick J, West J (2003) Why firms adopt open source platforms: a grounded theory of innovation and standards adoption. In: King JL, Lyytinen K (eds) Proceeding of the Workshop on Standard Making: A Critical Research Frontier for Information Systems. Seattle, Washington, pp. 236-257.

[14] Dedrick, J., West, J. 2007. Movement Ideology vs. User Pragmatism in the Organisational Adoption of Open Source Software. In: Kraemer, K., Elliott, M. Computerization Movements and Technology Diffusion: From Mainframes to Ubiquitous Computing. Medford: Information Today.

[15] Douglas Pitonyak A., OpenOffice.org Macros Explained, Hentzenwerke Publishing.

[16] Elhag HMA.,Abushama HM., “Migration to FOSS:Readiness Challenges”.

[17] Ellis, J, & Van Belle, J. 2009. Open source software adoption by South African MSEs: barriers and enablers. *Proceedings of the 2009 Annual Conference of the Southern African Computer Lecturers' Association,* 41-49.

[18] Fielding T. “Shared leadership in the Apache project”, Communications of the ACM 424, 38–39, 1999.

[19] 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.

[20] Floss Deliverable D18: FINAL REPORT, Part 2B: Open Source Software in the Public Sector: Policy within the European Union”, International Institute of Infonomics University of Maastricht, The Netherlands Berlecon Research GmbH Berlin, Germany June 2002.

[21] "Frequently Asked Questions about the GNU GPL" available from http://www.fsf.org/licenses/gpl-faq.html#MereAggregation; Internet; accessed on November 9, 2003.

[22] Gartner GW, 'How to avoid Pitfalls and Save Money with Linux Servers', Research Note. 17 June 2002.

[23] Ghosh, R. A., Glott R., Krieger B. and Robles G"Free / Libre and Open Source Software: Survey and Study"FLOSS,Deliverable D18: FINAL REPORT, Part 4: Survey of Developers

[24] Goode, S. 2003. Exploring barriers to Open Source Software Adoption in Australia's Top Firms: Implications and Directions for Research. *14th Australian Conference on Information Systems.* 26 – 28 November 2003. Perth.

[25] Gousios G., Karakoidas V., Stroggylos K., Louridas P., Vlachos V., Spinellis D. "Software quality assessment of open source software". Current Trends in Informatics: 11th Panhellenic Conference on Informatics, PCI 2007, volume A, pages 303–315, Athens, May 2007.

[26] Gritzalis S., Spinellis D., Georgiadis P. "Security protocols over open networks and distributed systems: Formal methods for their analysis, design, and verification". Computer Communications, 22(8):695–707, 1999.

[27] Hansmann B. "The Role of Nonprofit Enterprise", Yale Law Journal, 89, 835-901, 1980.

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

[29] Heffan I. "Copyleft: licensing collaborative works in the digital age", Stanford Law Review, Volume 49, No. 6, Pages:37, 1997.

[30] Henkel J. "F/OSS Software from Commercial Firms: Tools, Complements, and Collective Invention", GEABA Discussion Paper 02-27,2002.

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

[32] Hwang, S. 2005. Adopting Open Source and Open Standards in the Public Sector: Five deciding factors behind the movement. *Michigan Journal of Public Affairs*, Volume 2, Summer 2005.

[33] 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.*

[34] Juan M., Steinmueller E. "The F/OSS Way of Working: A New Paradigm for the Division of Labour in Software Development?", INK F/OSS Research Working Paper No. 1, SPRU-University of Sussex, Brighton, England, 2003.

[35] Kamseu, F., Habra, N. 2004. Adoption of open source software: Is it the matter of quality?

[36] Krishnamurthy S. "Cave or Community? An Empirical Examination of 100 Mature F/OSS Projects", First Monday 76, 2002.

[37] Kwan, S., West, J. 2005. A Conceptual Model for Enterprise Adoption of Open Source Software. *The Standards Edge: Open Season,* Sheridan Books, pp. 51-62.

[38] Lorraine M., and Patrick F., " How perceptions of open source software influence adoption: an exploratory study",. Proceedings of the 15th European Conference on Information Systems (ECIS 2007) (7–9 June, St. Gallen, Switzerland). St. Gallen, Switzerland: Universität St. Gallen, pp. 973–984.

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

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

[41] Mtsweni, J., Bierman, E. 2008. Challenges Affecting the OSS Adoption Rate in SA Government. FOSS4G 2008.

[42] Nagy, D., Yassin, A., Bhattacherjee, A. Organisational Adoption of Open Source: Barriers and Remedies. *Communications of the ACM,* 53(3), pp.148-151

[43] Olsoon C., and Ohrwall Ronback A., “ Collaborative Development of Public Information Systems: A Case Study of “Sambruk” e-Services Development”, eChallenges e-2010 Conference Proceedings Paul Cunningham and Miriam Cunningham (Eds) IIMC International Information Management Corporation, 2010.

[44] Osor, Guideline on public procurement of Open Source Software March 2010.

[45] Paudel, B., Harlaka., Shrestha, J. Open Technologies and Developing Economies. *CAN InfoTech, 2010*.

[46] Public Sector Forum, “Open or Closed?, A Survey of Open Source Software in Local Government”, August 2009.

[47] Rentocchini F., and Tartary D., “Open Source Software in the Public Sector: Results from the Emilia-Romagna Open Source Survey (EROSS)”, 2007.

[48] 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.

[49] Schmidt K., Schnitzer M. “Public Subsidies for F/OSS? Some Economic Policy Issues of the Software Market”, CEPR Discussion Paper, No. 3793, 2003.

[50] Singh V., Twidale M.B., and Rathi D, “Open Source Technical Support: A Look at Peer Help-Giving”, System Sciences, 2006. HICSS '06. Proceedings of the 39th Annual Hawaii International Conference , 04-07 January, 2006.

[51] Suzanne Scotchmer (2004) Innovation and Incentives. The MIT Press, London, England, 2004.

[52] Stallman R. “SCO, GNU and Linux”, 2005.

[53] Stallman R. “The GNU Operating System and the Free Software Movement”, in OPENSOURCES, supra note 2, at 53, 53–55.

[54] Varghese S., “Statskontoret - the Government's survey support”, Swedish Agency for Public Management, 2003.

[55] Ven, K., Verelst, J., “The Organizational Adoption of Open Source Server Software by Belgian Organizations”, Open Source Systems IFIP International Federation for Information Processing, 2006

[56] Ven, K., Verelst, J. & Mannaert, H. 2008. Should You Adopt Open Source Software?. IEEE – Software, 25(3), 54-59.

[57] 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, 116-128.

[58] Weber S. “The Political Economy of F/OSS Software”. BRIE Working Paper 140, Economy Project Working Paper 15, 2000.

[59] Weber S. “The Success of Open Source”, Harvard University Press, 2004.

[60] Wheeler D. A. ,”How to Evaluate Open Source Software / Free Software (OSS/FS) Programs”, August 2011.

[61] Wong K, Sayo P., “ Free / Open Source Software. A general Introduction”, IOSN International Open Source Network, 2004.

[62] Yuan, E. 2009. Adoption of Open Source Software by Singaporean Companies. *Queensland University of Technology*.

APPENDIX A – ABBREVIATIONS

Abbreviation	Explanation
FOSS	Free and Open Source Software
OSEPA	Open Source Software Usage by European Public Administrations
GPL	General Public License
IOSC	Internet Operating System Counter
FLOSS	Free / Liberty Open Source Software
PA	Public Administration
LAMP	Linux operating system, Apache web server, MySQL database and PHP scripting language
API	Application Programming Interface
CSCW	Computer-Supported Cooperative Work
IT	Information Technology
OSI	Open-Source Innovation & Open Source Initiative
VBA	Visual Basic for Applications
IIS	Internet Information Server
HTML	HyperText Markup Language
W3C	World Wide Web Consortium
IPR	Intellectual Property Rights
ERP	Enterprise Resource Planning
OS	Open Source
FDI	Foreign Direct Investment
SME	Small and Medium Enterprise
BSD	Berkeley System Distribution
GUI	Graphical User Interfaces
EULA	End-User Licence Agreements
SW	Software
DOI	Diffusion of Innovations

FS	Free Software
OSS	Open Source Software
TCO	Total Cost of Ownership
ROI	Return On Investment