Study on the effect on the development of the information society of European public bodies making their own software available as open source

Final report

Authors: Rishab A. Ghosh (MERIT)
Rüdiger Glott (MERIT)
Patrice-Emmanuel Schmitz (UNISYS)
Karsten Gerloff (Merit)
Kamini Aisola (UNISYS)

December 12, 2006

Disclaimer:

The opinions expressed in this study are those of the authors and do not necessarily reflect the views of the European Commission.

Table of Contents

1 Opportunities analysis and barrier report....5
1.1 Background, objectives and approach of the study....5
1.2 Incidence and role of OSS in European local governments....7
1.3 Challenges for public bodies....8
1.3.1 Programming capacities....8
1.3.2 Skills and open source software....9
1.3.3 Legal aspects....9
1.4 Advantages of publishing software under an OS licence....10
1.5 Conditions and effects of publishing software under an OSS licence – theoretical considerations....12
1.5.1 Positive marginal costs for software....12
1.5.2 Producers....14
1.5.2.1 Effects on R&D spending....14
1.5.3 Users....16
1.6 Conditions and effects of publishing software under an OS licence – empirical evidence....19
1.6.1 Presentation of cases studied....19
1.6.2 Findings from the survey....22
1.6.3 Organisational issues....24
1.6.3.1 Sound management, e.g. through a technical committee....24
1.6.3.2 Maintaining quality, including security, user-friendliness and documentation....25
1.6.3.3 Programming and maintenance capacities (“readiness”)....25
1.6.4 Economic and financial issues....26
1.6.4.1 Spill-over effects....28
1.6.5 Reliability, extensibility and security issues....29
1.6.6 Technical issues....29
1.6.7 Legal and regulatory issues....30
1.6.8 Effects on government services....31
2 Report on possible support for the deployment of OSS....32
2.1 Aspects of a successful Open Source Support strategy....32
2.1.1 Motivation / knowledge sharing....32
2.1.2 Assessment....33
2.1.3 Strategy definition....33
2.1.4 Strategy execution and evaluation....33
2.2 The role of and need for OSS competence centres....33
2.2.1 Licence questions....34
2.2.2 Statutory questions....34
2.2.3 Procurement questions....35
2.2.4 Providing Internet tools and collaboration services....35
2.2.5 Project/product issues....36
2.2.6 Existing repositories and competence centres....37
2.3 The need for an OSS community....37
2.3.1 Developer's motives....38
2.4 Possible support roles for the software industry....39
3 Guidelines and practical tool kit for making publicly owned software available under an open source licence....41

3.1 Scope of “Public sector software”....41
3.2 Reasons and base for public software and resource pooling....41
3.2.1 Pressures for efficiency....41
3.2.2 The policy of building the knowledge society....42
3.2.3 The expected impact on society....42
3.2.4 Is common development (mutualisation) an option?....43
3.2.5 Where could Open Source help?....43
3.2.6 Where to find best practices, examples or guidelines?....44
3.3 The selection of a licence....44
3.3.1 Licence types....44
3.3.2 Guide for choosing a licence....46
3.3.3 Which licence is most appropriate for the public sector?....47
3.3.3.1 Dual licensing....49
3.3.3.2 Copyright dispersion issues....49
3.3.4 Would a EUPL be better adapted to European realities?....50
3.4 Check list before publishing Public software....51
3.4.1 A general need and commitment....51
3.4.2 Ownership....52
3.4.3 Suitable disclosure and publication....52
3.4.4 Developed according to the open source model....52
3.4.5 Suitability for licensing....53
3.4.6 Suitable for obtaining long term support from a community....55
3.5 Barriers and questions on distributing Public software....56
3.5.1 Public assets disposal....56
3.5.2 Competition with the industry....57
3.5.3 Confidentiality requirements....57
3.5.4 Procurement requirements....58
4 Conclusions from the study: Characteristics and impact of OSS releases by public bodies....59
4.1 General characteristics....59
4.2 Organisational challenges....60
4.3 Personnel / skills requirements....60
4.4 Economic aspects....61
4.5 Technical challenges....61
4.6 Legal aspects....61
4.7 Impact on government services 62
4.8 Impact on the information society....63
4.9 Considerations for a successful Open Source strategy....65
Glossary....67
References....70

1 Opportunities analysis and barrier report

1.1 Background, objectives and approach of the study

In the background of releasing public software as open source is a great hope to obtain a "multiplier effect" (a developers community providing improvements for free) just as it was obtained with GNU/Linux or with the Wikipedia experiences. But is this hope grounded and will the expectations finally be satisfied? Is the impact real both in quality and in quantity? And will there be a similar multiplier effect on the economy and the Information Society at large, in terms of participation, ICT adoption, growth and innovation, competition, interoperability and transparency? The core task of the project is to examine if this "multiplier effect" is a genuine probability, under what circumstances it would occur, and what the impact of such possible development on the take-up and growth of the Information Society would be, both qualitatively and quantitatively.
In order to provide meaningful answers to these questions that help to advance the public sectors in Europe, the study

- carried out quantitative and qualitative analyses of the possible impact on the development of the Information Society and present ICT take-up scenarios, based on realistic data, using indicators for the level of ICT adoption, and explaining the likelihood of the scenarios;
- examined
  - if there is a possible effect on interoperability between public organisations and how this effect could be defined,
  - if there is a potential spill-over effect to businesses and citizens,
  - if the publication of software used for processing citizen and business data would contribute to the objective of transparency in government
  - if there is a kind of software that is typically owned by public bodies,
  - which type of software would have a higher potential impact if published using an open source license,
  - what the volume of software would be that could be made available
  - and quantified the specific advantages of publishing software of public organisations under an OSS license
  - if there is any saving of costs or gain in functionality, quality or efficiency to be expected
  - how software would need to be published in order to be useful to others
  - if there is a market, innovation and diversity impact to be expected and how this could be characterized; and
- clarified whether the rationale for and principles of making public information available apply to the publication of software fully owned by public bodies.

The study investigated this through a combination of empirical, quantitative and qualitative approaches, supported by economic modelling. For the study existing data on OSS in European public sector has been re-evaluated as well as new data on software release activities of public bodies has been generated. The project consortium used a variety of publicly available databases as well as data gained in previous research projects and in surveys (on public organisations, private sector firms, and citizens), expert interviews (with IT managers / developers in public organisations, individual developers and firms providing Open Source Software for public organisations, other public and private organisations that support Open Source Software development on the local and regional level) and 6 case studies (of public organisations contributing to Open Source Software)

carried out in the context of the proposed project. A survey revealed information on OSS activities of European local governments for 220 organisations in 9 European countries (see the annex for a detailed description of the purpose, methodology, and sample of the survey).

The aims of the quantitative analyses are to evaluate the intensity to which public organisations contribute to the development of the Information Society by providing Open Source Software, to identify strengths and weaknesses (and possible future potential) of public sector organisations on the markets for software, and to evaluate to which degree public organisations can complement the Open Source activities of private sector firms. For these purposes, quantitative survey proposed for this project, combined with existing data available to the consortium and described in the table of Data Sources shall:

- provide estimates of the share of public organisations owning software that could be distributed under an open source licence, and the share that is distributed under an open source licence
- compare the development of Open Source Software in public organisations to Open Source development activities in the private sector
- identify typical sectors (applications) of the markets for software in which public organisations are active and those that are not (yet) covered by them

Qualitative research methods, i.e. expert interviews and case studies, were used to

- identify the proponents and opponents of an engagement of public organisations in contributing Open Source Software
- identify relations to other public and private organisations, support organisations, the FLOSS community, educational institutions, and policy makers (this tends to take place mainly on the local and regional level, though we will investigate also national and international relations)
- reveal the reasons for public organisations to participate in this field
- examine the concrete legal, economic, and organisational conditions under which public organisations can successfully participate in the provision of Open Source Software and which difficulties must typically be overcome in this process
- estimate to which degree and in which directions public organisations will contribute to Open Source Software in future
- identify how development of Open Source Software changes procedures of existing government activities and creates new government activities
- evaluate cost and benefits of Open Source Software contribution for public organisations
- find out to which degree and for which purposes public organisations interact with other public and private organisations and with citizens in the context of their Open Source Software activities
- evaluate improvements of interoperability and transparency of government activities through contributions to Open Source Software
- identify best practices of developing and publishing Open Source Software in public organisations

The case studies that have been analysed for this study are:

- The Council of the London Borough of Camden, which developed and released APLAWS, a Content Management System for public authorities
- The Nederlands Forensisch Instituut (NFI – Dutch Forensic Institute), which developed and distributed TULP2G, a program that allows reading out the content of mobile phones
- The Beaumont Hospital in Dublin, which migrated largely towards OSS and began to develop and to distribute health care-related OSS systems in the course of this process

- The Dutch public broadcasting company VPRO, which developed MMBASE, a Content Management System that is specially designed to handle multimedia-content of all kind
- The provincial government of the Spanish province of Extremadura, which realised the LinEx project, a localised version of GNU/Linux originally developed for the educational sector.

Based on the results of the quantitative and qualitative studies, combined with the economic modelling, the project consortium develops scenarios for the impact of public organisations' contributions to Open Source Software on the development of the Information Society. Whether and to what degree to Open Source software from public organisations contributes to standard general purpose application domains, embedded in the wider FLOSS community, or to what extent they contribute to niche application domains and rely on in-house capacities, is among the starting points for these scenarios.

1.2 Incidence and role of OSS in European local governments

The survey on local governments that was carried out for this study revealed information on 220 public bodies in nine European countries. 217 local governments provided an answer to the question whether they own software that could be distributed under an Open Source license. It turned out that only 10 per cent of the respondents own such software. About 54% of this software is "general purpose software" (that is of use to anyone, e.g. operating systems, office applications, etc.), 45 per cent is "specialised software", i.e. software that is useful for Public Administrations but also other organisations, such as content-management systems, collaborative workflow and calendar systems, hospital management systems, security software, etc. 27 per cent answered that they own "software useful only for the public sector" (such as land records management, court management, internet tax declaration etc.).[1] 38 per cent of the governments that own software that is distributable under an Open Source license stated that they developed this software themselves, while 33 per cent reported that they have contracted somebody / a company to develop this software for them, and 19 per cent combined these two opportunities (one local government could not answer this question); 10 per cent did not know who the software developed.

These figures indicate a considerable discrepancy between the role public organisations play as users of Open Source Software (which is high, compared to private sector companies) and their role as producers of such software. How these figures must be interpreted is an open question at the moment because there are no comparable figures for private sector companies. Moreover, whether these contributions of public organisations to Open Source Software derive from strategic plans of these organisations or from individual and voluntary work of one or more employees of these organisations cannot be answered at the moment, either.

In the 2004 FLOSSPOLS survey of 955 public authorities across 13 EU countries, over one third of FLOSS non-users and over 41% of FLOSS users customise software regularly or often. As shown in Figure 2 below, almost half (45.6%) of regular customisers also use external maintenance services regularly or often. It is likely that these public bodies do not own their customised software.

However, 15% of respondents customise regularly or often, but use external maintenance services sometimes or never. Of these, 61% are FLOSS users, representing 9% of the total sample of public bodies. These are likely to both own their customised software (as they rarely use external maintenance services) as well as be releasing, or ready to release it as open source (being FLOSS users). Indeed, the 9% figure is close to the figures for owned software and open source published

1These figures sum up to more than 100% because the respondents could check more than one answer to this question, see question number 1a (Q1a) in the questionnaire in the annex.

software for Dutch public bodies.
The FLOSSPOLS government survey revealed that awareness of FLOSS leads to increased use of FLOSS, disregarding whether or not an organisation actually uses FLOSS. Unaware users of FLOSS show stronger reluctance to FLOSS than those who know clearly what systems are used in their organisation and who have a clear attitude towards or against FLOSS. A main obstacle for the usage of FLOSS are fears concerning lack of technical support for FLOSS systems and high training costs in the course of migrating towards FLOSS systems in the organisation. Apart from these fears, vendor lock-ins and business-cycles played also an important role as obstacles to the use of FLOSS.
Vendor lock-ins may occur as an issue of procurement policy if in public tenders competition is limited through either naming platforms to which new software must be compatible or by explicitly naming proprietary software systems. Budget cycles can also become a hindrance for FLOSS usage because a good part of migration costs (such as training costs) occur in the first year of the migration process, whereas savings tend to occur later in the course of the migration process. Since budgets are usually calculated for each year it is difficult for the IT department to convince other departments from the usefulness of a migration. A solution to this problem could be to schedule budgets for a period of three years, which would probably ease migrations to FLOSS.

1.3 Challenges for public bodies

The main challenges posed to public administrations by OSS concern skills, legal aspects and interactions with the FLOSS community.

1.3.1 Programming capacities

An important precondition for public organisations to contribute to the provision of Open Source Software is to have or build up own programming capacities, which can be reached by employing skilled in-house staff as well as by contracting external firms or individual developers. Both strategies result in additional cost (relative to pure software and IT administration or pure software usage), at least in the short to middle term, and therefore affect the budgets of public organisations. The results of the OSOSS surveys (Data Source S5) as well as of the FLOSSPOLS local governments survey (Data Source S4) indicate that staff in IT departments of public organisations are characterised by administrative skills more than by programming skills.
We believe that a significant contribution of public organisations to Open Source Software can probably only be reached among public bodies that see their IT departments as not limited to administrative support services, but as a strategic resource for the organisation or even beyond, at the local/regional/national/community level. This would require fundamental changes of the organisational structures of public organisations as well as of the skills and self-perception of IT managers and of the procedures in these organisations – this was the case, say, in Extremadura.

1.3.2 Skills and open source software

The FLOSSPOLS survey shows that the availability of skills is clearly perceived to be a problem for open source use. However, the same data show that this may be a problem of perception rather than a fact: the perception is strongest among FLOSS non-users and those who are unaware of FLOSS. FLOSS users, who must actually bear the costs of training and technical support, tend slightly to disagree with the statements that training and support are expensive or hard to find. A parallel survey of employers showed a clear market impact of these skills, which were seen by potential employers (HR managers in ICT-intensive firms in Belgium, Germany, UK, France, Spain) as economically valuable.

1.3.3 Legal aspects

Public authorities may be concerned with copyright and patent issues in relation to their use / development / publication of open source software. Most PA now ask from their contractor the "property" of the code developed by these contractors for the PA. However, if such property includes the right to re-distribute the question arises whether or not the PA is covered against copyright or patent suits.Open Source redistribution makes the code visible to the public and could generate suits that would never happen if the software was maintained inside the protected PA perimeter.

If most PAs use some open source software, the decision to "go to Open Source distribution" by publishing the source code and distributing both the source and the object code is still very rare. Some PAs do not see the necessity of software code distribution. In case a real distribution of software is decided upon, the choice of a licence is fundamental. Investigation has shown that the decision to distribute was generally taken without knowing all conditions and forecasting all the impact, which can be considerable.

Contracts with providers should not only assign to the PAs the ownership of the code, but explicitly foresee redistribution and cover the PA's responsibility in the case of redistribution. Such contracts should impose a specific documentation of all solution components licensing conditions and clarify the approval process of these conditions by the PA: indeed, building a solution with various existing components is now a normal way of working. The newly developed software (which would be fully "owned" by the contractor and thus the PA) will therefore probably be unusable without the pre-existing software components written by others (which could be owned by the contractor and the PA only notionally, under terms of the licence under which the pre-existing software is distributed).The used components may be licensed under various licences (proprietary or not) and even when all components are "open source" according to the OSI definition, their licences may be incompatible (e.g. if two different copyleft licences are used). In such cases, the PA may well receive from its contractor the "ownership" of the developed solution (as far this is in conformity to the various components licences), but any redistribution would not be simple nor efficient: because of licence conflicts, the solution might not be possible to distribute under a single licence.

A wider software distribution by PAs cannot be separated from the need to provide an appropriate coverage of liabilities – in particular in the case of copyright or patent infringement. Patent infringement is currently seen as the one of the main threats to open source software distribution: PAs perceive some ICT companies as "not producing anything but just patenting things" and the simple fact of reproducing (even with a new specific source code) an idea or an approach to solve an ICT problem could constitute a patent infrigment. Suing a PA and becoming "visible" is perhaps easier and potentially more profitable than suing a member of an informal community and this makes any responsible body think twice prior to release anything under an open source licence. If this ever happen, it would probably not stop a PA from *using* open source, but it could prevent it

distributing its own software[2].

1.4 Advantages of publishing software under an OS licence

At present, Open Source Software production seems to be dominated by individual developers and private sector companies, although universities have a significant contribution (see Figure 1). However, the FLOSS survey showed that as early as 2002, Public sector institutions played an extremely important role as users of Open Source Software, in some areas even exceeding OSS usage rates of IT-intensive industries. (Wichmann 2002a, p17). The FLOSSPOLS survey of developers, among other sources, show that growing interest in and experience with Open Source Software increases the likelihood of contributing to Open Source Software development; so we believe it is likely that there is a great potential for public organisations, as leading users of open source, to play an emerging role as providers of Open Source Software with a significant impact on the development of the information society.

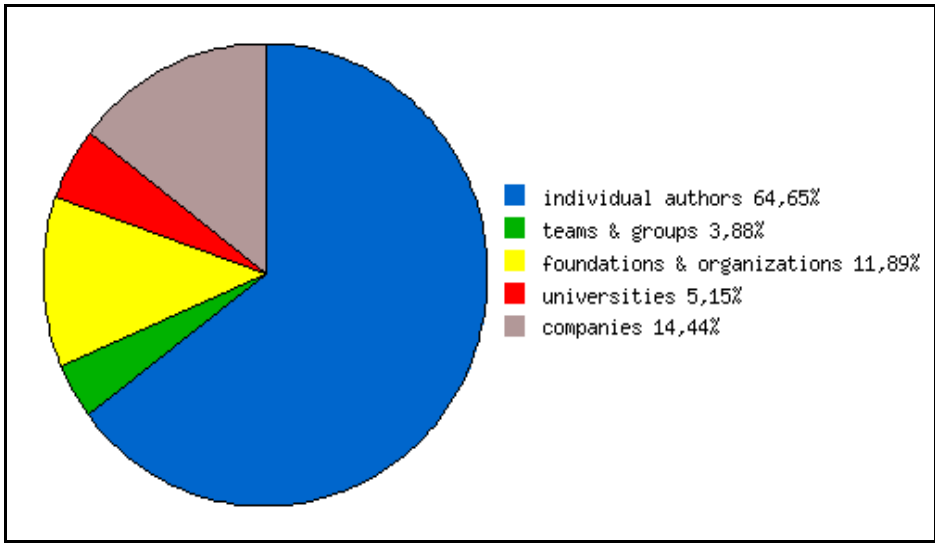


Figure 1: Distribution of FLOSS code productivity by individuals, firms, universities

Source: URJC, data available at MERIT. Shows code contribution share for Debian 2002.

What this impact would be can partially be inferred from the reasons that made the usage of Open Source Software attractive for public organisations. They range from cost-effectiveness (Robert Francis Group, 2002; Winslow 2004; Bayerischer Oberster Rechnungshof, 2001, pp63-73) and improved flexibility (Varian & Shapiro, 2004, pp2, 11); increased long-term control over maintenance and support; increased technical features, increased interoperability, and independence from software vendors (KBSt, 2003, p19); to security aspects (Deutscher Bundestag, 2001) and improved reliability (Wheeler, 2002). The FLOSS user survey (Wichmann, 2002) showed that IT managers in the private and public sector considered higher stability and and better performance as even more important advantages of open source software than licence costs savings.
Public organisations making their own software available under Open Source licenses will contribute significantly to the pool of cost-efficient software of potentially high quality (and likely

2 Note that PAs, especially as users, may not be under threat for patent infringement by using open source, and in any case the threat for PAs with open source is not significantly different from the threat of using any other form of software, as discussed in Ghosh, Rishab; Bakels, Reinier; Schmitz, Patrice-Emmanuel, 2005: "Patents and open source software - What public authorities need to know", published by the European Commission (IDABC), available at http://europa.eu.int/idabc/servlets/Doc?id=20675

high attention to user requirements, having originated in customised software for specific needs). This provides, first, a pooling of resources among public bodies potentially allowing significant cost savings as they are able to reuse software applications made for specific public sector requirements, rather than duplicate them for each public body individually.
Second, as Varian & Shapiro (2003) exemplify with Linux, there are farther-reaching reasons for the public sector to adopt open source software: “Public sector technology managers have additional reasons to adopt Linux. Adoption of Linux platform software promotes the training of software engineers and provides an open platform on which commercial or open source applications can be built, thereby spurring the development of a robust domestic industry. Certainly, any government information technology manager seeking to put in place a flexible computing environment that also helps promote the domestic software industry should give serious consideration to Linux.” (Varian & Shapiro, 2003, pp20). Some of this conclusions are also supported by data from MERIT’s previous research, including on the skills training of software engineers (described below in the section on skills) and with better empirical support through MERIT’s study of the regional economic impact of the support of open source, including release of open source software, by Extremadura, Spain (Data Source S8). The sustenance of a further software development ecosystem through the release of open source software by public bodies is a particular feature of the economic modelling in WP4.
Cornelia Kutterer of BEUC, the European Consumers' Organisation,[3] was asked to provide the consumers' perspective on PAs making their own software available as FLOSS. She strongly supported view that economy and society – and thus citizens and consumers - would benefit from an increase of the share of FLOSS in the market for software: "OSS production in public administration is not a core issue of BEUC, but BEUC has a clear opinion towards this subject and thinks that there would be indirect positive effects for consumers if the public sector would play an active role in the provision and usage of OSS: In general, OSS helps to improve competition because it provides better products than proprietary software and it avoids vendor lock-ins. As a consequence, consumers would be provided with better quality and liberated from the market power of large software vendors. Governments and public administration should support usage and development in the public sector in order to increase network effects and interoperability. This does not mean that proprietary software should be replaced by OSS, but that OSS should generally be offered to the consumer to the same degree as proprietary software in order to interact with public administration.”

1.5 Conditions and effects of publishing software under an OSS licence – theoretical considerations

In this section we develop some theoretical concepts which can be used to analyse the conditions under which public release of open source software might have strongly positive impacts on the software sector as a whole. The main issues revolve around costs and benefits of software development and use. The interplay among different costs and benefits is highly complex. Our goal here is not to make innovations in modelling of industrial structure and innovation. Rather it is to use existing, well-established models, and the results from them, to shed light on the question at issue.
The theoretical structures developed here are based on two standard models of industrial

3 BEUC has long participated in the political discussion on FLOSS and repeatedly pointed out that OSS provides positive effects on competition and growth and that there are severe risks aligned with the dominant market position of proprietary software. This subject remains an important topic for consumers' organizations and they will go on playing a role in this field. Most recently, the Trans Atlantic Consumer Dialogue (TACD) has discussed user perspectives on Open Document Format (see http://www.tacd.org/docs/?id=304).

organization, namely Cournot (1838) and Hotelling (1929) models. These two structures are applied to the specific case of the private software sector and, in our opinion, they capture very well several of the key factors at stake. In particular both models describe a situation where the market is oligopolistic, which means that relatively few producing firms with high market power are present. Both also assume that marginal costs are positive and that they make up a non-trivial part of the total costs.

1.5.1 Positive marginal costs for software

Before we go on, it is worth commenting, since this assumption seems to violate one of the received pieces of wisdom about software, namely that it has very high fixed costs and very low marginal costs. The former stems from the high development costs of any sophisticated piece of software, which is taken as the cost of the first unit. The cost of the second, and further units are merely the cost of electronic duplication (virtually zero) and then installation. Here, "production" indicates the duplication and installation of a fixed piece of software code, representing a single version of one product.

But there are two problems with this traditional assumption. First, it only holds for packaged software. For custom software and services, which according to our research makes up around 50% of the market, the cost structure is quite different. Since software firms are not selling the exact same solution to all customers (and often not even a similar solution, due e.g. to contractual restrictions on code reuse), they indeed face substantial positive marginal costs for each customer.
The second problem with the traditional assumption of zero marginal cost for software is that it focuses on the wrong unit. Producers of packaged software are likely to present marginal positive costs as well. This is due to the strong commitment that producers must make to customers if they want to reap the profits of network externalities' advantages. A user decides to adopt particular software if and only if he foresees a commitment from the producer to maintain and upgrade it up to some time in the future. In this sense, if producers want to exploit both network and lock-in effects they are likely to assure the upgrade and maintenance of the product, and from this perspective, the "second unit" can be seen as the second version, rather than the second copy of the first version.[4]

Certainly, within a single version of a piece of software, marginal costs are very close to zero. This does mean that firms can capture and exploit a market for a single version. But if this is the only thing a firm does, it faces a problem for the future. without further effort it will not attract the next generation of users. In addition, though, the firm will lose its original customer base, as hardware and particularly operating system software are upgraded, making the original version of the software obsolete or literally un-usable. Consequently, most successful packaged software goes through multiple versions, apparently endlessly, in the strongest cases. The presence of network externalities, technological interdependence (between hardware and software), and consequent lock-in effects on the users' side is a strong argument in favour of an intense commitment from the producer side as well.

This is not the typical way that software as a good is characterized, but by using this characterization we can deploy well-understood economic models to capture the effects that matter

4 In essence, what we are suggesting is that if a firm sells n copies of version 1, it is not the costs of units 2 to n which matter, but rather the cost of the n+1th unit, namely the first copy of version 2, which is crucial. From the point of view of social welfare, it is that cost which is vital, and which might be affected by the release of publicly-developed open source. Treating a "version" (rather than a copy) as a unit of the good permits us to use standard assumptions about marginal cost, and thus use standard models from industrial organization theory.

in this case. Hence, under the reasonable hypothesis that PAs are important customers of private software firms this points out the strong commitment software firms should make towards long term relationships with these customers. In our surveys, from 44% to 67% of respondents see their organization as highly dependent on suppliers. Furthermore, most of governments rely on less than 4 suppliers, meaning that their dependency is high and that it reflects the oligopolistic structure of the market assumed in the theoretical model.

All these factors help to explain the nature of the marginal costs. In fact, they put pressure on costs, in particular marginal ones. So, not only the first unit of software is expensive to produce (as the standard economic theory asserts) but also following units will be have non-zero costs as it must be incorporated in the single unit the cost of maintenance to be provided – especially given our perspective in this model that treats the software version as the "unit".

1.5.2 Producers

Thus, assuming small but positive marginal costs, the standard Cournot model, which has been usually used to study traditional sectors, can be extended to the analysis of software.

The main hypotheses we use are:

1. The number of firms is fixed. This assumption will be relaxed in the discussion where we consider entry.
2. There are several (but relatively few) firms producing similar, but not identical, products. This implies that firms have a certain degree of market power. In addition, since products have similar characteristics users can easily substitute among them without incurring a decrease of their level of utility. This is a rather strong assumption that will be relaxed once that Hotelling model is introduced.
3. Firms do not co-operate. This means that firms are not allowed either to co-operate, for example by setting a market standard, or to collude, for example to prevent entry into the market.
4. Firms play a two-stage game where they first choose simultaneously the quantity to produce and then, knowing each other's capacity, they simultaneously choose price.

This describes a standard model, well-known in the industrial organization literature, namely the Cournot model of oligopolistic competition. Here we highlight several results that are useful for our investigation:

- The prevailing market price is lower than the price obtained in monopoly and the aggregate profit is lower than monopoly profit;
- The prevailing market price is higher than the perfectly competitive price and the aggregate profit is higher;
- As the number of firms in the market increases, both market price and each firm's profit decrease. At the extreme, as the number of firms gets very large, the market approximates a competitive market;
- The market share of any firm is negatively related to its marginal cost of production.

In order to understand the dynamic followed by market shares of firms in the software sector and the rate of competition and innovation of the sector it is important to understand factors affecting

marginal costs of production of these firms.

The costs or production for any firm can be seen as affected by three main factors:

- Technical difficulty in producing the software
- Competence of the producing firm
- Public knowledge available to the firm

1.5.2.1 Effects on R&D spending

Our concern is with the effect that the release of FLOSS code held by public organizations will have on the profitability and the competitiveness of firms operating in the European private market. To begin we ignore the possible effects on entry to the market, and look only at the effects on existing firms, assuming that no new firms enter as a result of OSS release. Under this condition, the release of OSS is likely to change the variance in the cost structure of firms. After release, all firms will have access to a larger stock of public knowledge. All else equal, this implies that they have a more similar mix of inputs than they had before the release. This implies a reduction in the variance of their costs.
Since market shares are determined by costs, according to the Cournot model, and profits are determined by market shares and costs, we observe a follow-on effect of reducing variance in the profits of firms. Market share, market power, and profits of the large firms will fall, those of the small firms will rise. Where this has a dynamic effect is in R&D expenditure. In the software industry of established firms, R&D spending tends to be financed by retained profits, and indeed there is evidence that firms tend to spend a fixed proportion (around 15%) of their profits on R&D.

Decreasing market shares will mean decreasing profits for larger firms, and hence decreasing resources to be devoted to R&D activity. The opposite happens for smaller firms that have more money for the R&D process.

A shift of profits from high-profit to low profit firms will, all else equal, improve the industry-wide returns to R&D, and so increase industry level innovative performance. But so far we have ignored the effect on total industry profits. With a decrease in the variance of costs, total profits are likely to fall. Thus there are two counter-valing forces: reduced variance in costs reduces the variance of profits, and thus R&D and so improves industry R&D performance; but reduced cost variance lowers industry profits, reducing industry R&D, and so reducing innovative output. Which effect dominates cannot be judged a priori. However, the effect is most likely to be positive in sectors in which the variance in firm size is large, particularly if the largest firms are very large and so well into severe decreasing returns to R&D spending.[5]

So far we have also assumed that no firms enter the market in response to an increase in public knowledge. If this is relaxed, and entry into the market is allowed, other effects come to play an important role in order to understand the possible scenarios following a release of Open Source code by public bodies. In this regard, the nature of the firms entering is extremely important. If the majority of firms entering are large companies then no efficiency gain will probably take place, while with small firms entering the efficiency gain will be higher. At the same time, regardless of the size of the firm, the entrance by new firms into the market is likely to produce a “threat effect”

5 Anecdotes from the software industry in the 1980s and 1990s describe the R&D spending by Apple and Microsoft as “burning money” due to the huge amounts spent. If true, this was a case in which a reduction in variance of profits among firms would certainly have improved industry innovation performance.

on incumbents. This is due to the fact that they fear new entrants since more the presence of more firms tends to reduce the profits of existing firms. Thus, incumbents are likely to reduce dead-weight losses due to monopoly power by themselves in order to compete effectively with new entrants (Gilbert and Newbery, 1982). Thus the threat of entry, initiated by the release of OSS code, will reduce monopoly profits, and constitute an immediate gain for the consumer. However, the reduction of profits may imply a reduction of R&D, creating a longer term slowdown in the rate of innovation. Again, it is exceedingly difficult to weigh the balance of these effects a priori.

Finally, we must address one dynamic aspect of R&D. R&D has two functions: it provides immediate innovation; and it provides learning on which to base future innovation. A reduction in R&D by a firm today, may reduce its ability to undertake productive R&D tomorrow. Again, this effect is likely to be governed by decreasing marginal product (it becomes harder and harder to learn, the more learning one does) but this may operate differently than the decreasing product in the innovation aspect of R&D, and so there may be a tension, again, between the short run and the long run However, while this "learning-to-do R&D" effect no doubt exists, it is likely to be significantly smaller in magnitude than the previous effect of decreasing marginal product in innovation, particularly in a field such as software, where all firms that survive must have high abilities at R&D to begin with.

According to this dynamic interpretation, the increase in the public knowledge input consequent to the release of OS code by public bodies presents a trade-off. On one side, the availability of this knowledge stock base for free fosters innovation. On the other side, the effect on profit structure is likely to diminish the R&D spending both at the firm and at the industry level reducing the rate of innovation.[6]

It is important to point out the difference between R&D spending and innovation. A reduction in R&D spending may not necessarily reduce innovation if, for example the reduced spending is used more productively, or if there is a bigger platform of public knowledge which can be leveraged by private firms. (Indeed one could argue that the presence of a larger public knowledge base makes private R&D spending more productive.) Similarly, increases in R&D spending may not increase innovation, particularly if that innovation relies heavily on publicly available knowledge.

1.5.3 Users

We acknowledge that not all software users have the same tastes, or want or need to do the same things with their software. Indeed users can be very heterogeneous, and one of the arguments against standardization is that any dominant standard will fail to serve some consumers well. How well users are served by the software products available is the focus of the next section.
To account for these heterogeneous tastes and needs, we use the horizontal differentiation model. Its main implication is that each firm differentiates itself and its software product from its rival in order to soften price competition and obtain some sort of monopoly power.[7] Different products satisfy

6 For an analysis on the relationship between market structure and R&D spending in a dynamic setting see Cowan (2002). There, today's R&D productivity is formally related to its future's productivity. The final result underlines how, at the firm level, the number of firms in the industry are negatively related to R&D. On the contrary, the relationship is not so clear at the industry level where it can be both positive and negative depending on specific parameters (rate of knowledge accumulation, success probability of innovation processes, etc).

7 In the localization interpretation, each firm decides to locate enough far from its rival in order to avoid price competition. Instead, in the product differentiation case, firms produce goods as differentiated as possible.

different users' demands

Each software product comprises a unique set of features. For the sake of simplicity, suppose we can represent these features as being at a point on a line.[8] For example, in Figure 2 the *x* axis represents product space, i.e. the different types of software products available in the market. Different products satisfy different consumers' tastes.[9] Obviously, points close to each other represent products that would be tailored to similar preferences.

On the *y* axis the utility for the consumer is shown. Assume that only a few firms are producing software products. If a consumer is located at the same point is space as a firm, the consumer receives high utility from the product. On the other hand if the users is far from any firm, he must select one product, then either expend resources customizing the software, or simply be dissatisfied with the performance of the product, but have to live with it. Suppose there are three firms, located at A, B, and C. A user located at a will choose the product of firm A, and receive utility *u1*. The tent-shaped straight lines represent the utility from a particular product received by a user at any point on the line. Obviously, firms will produce products whose characteristics overlap which means that they will compete for consumers that are interested in them. This case is graphically displayed by the portion of x axis under crossing lines, e.g. segment bc.

If we assume that a software sector has few producing firms in it (A, B, and C) as the one presented in the figure, then the variety of products present in the market and, as a consequence, the level of utility for consumers can surely be increased by the entrance of new players. As we argued above, publicly available knowledge or inputs reduces costs. It equally reduces barriers to entry. Thus release of code would encourage firms to enter this market. They would enter at D and E.[10] By entering they increase the level of competition, since there are more segments on the product space that are located under intersecting lines. Before the release of the code, a consumer located at a purchased good A, and realized a level of utility equal to *u1*. After firms D and E enter the market, that consumer switches to good D, and his or her utility rises to *u2*.

8 If the software had one feature: calculating square roots, for example, a point on the line might represent the ratio of accuracy to speed of calculation. Different users have different preferences for accuracy over speed, so they too would be located at a point on this line.

9 Many different types of products are present in the software market. A main distinction in the classification is between software packages and customized software and services. Of course, the variety in users' tastes is easy to understand in the practice: some users prefer quite standardized products for normal operations (operating systems, enterprise resource planning, payroll/personal management, etc). On the other side, some other users have particular needs that are satisfied through custom software (custom software development services, systems implementation, etc).

10 New firms position exactly in between other firms. In fact, this is the only way to maximize the monopoly power they have on a part of the product space. Indeed, this is the main result of the Hotelling model.

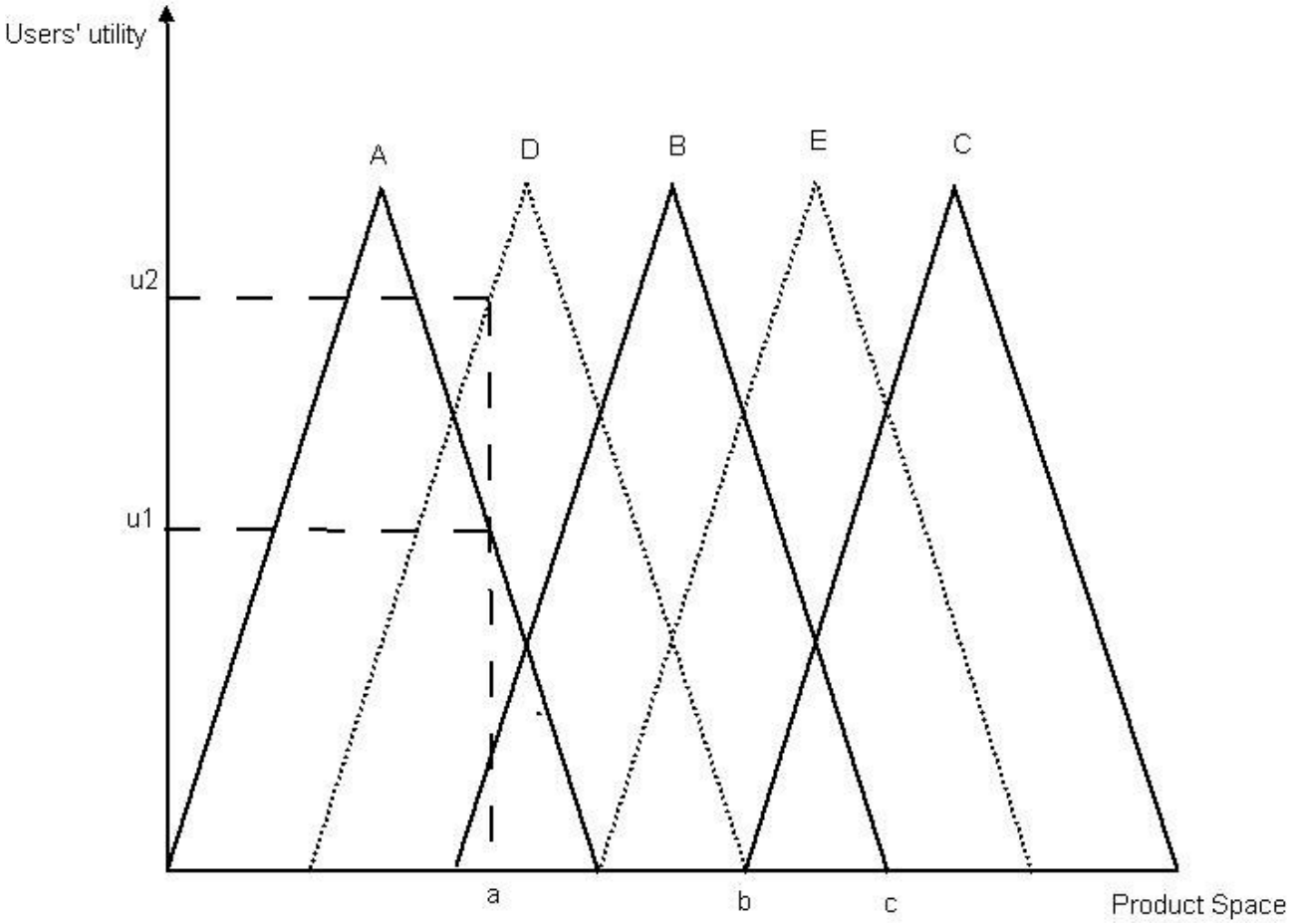


Figure 2: Effect of code release by Public Administrations in a horizontal differentiation setting.

A more careful interpretation of Figure 2 leads us to see when this effect is likely to be strong. The entry of new firms (D and E) is valuable because some users (such as a) are better able to have their needs fulfilled by one of the new products. Thus what is crucial in determining the size of this effect is not, in fact the number of incumbent firms, but rather how the location of the incumbent firms, in technology space, relates to the location of users in that space. The Figure implicitly assumes that users are evenly spread over the line, so an entrant at D will serve many users well. But if users are in fact bunched in their needs, and are mostly located at directly under A, B, and C, an entrant at D will have only a small impact.

This suggests that lowering entry costs by releasing software as OSS is most likely to have a big effect on a market segment in which there are few firms, but many users, and more importantly, many different types of users. Niche market software is not very likely to satisfy this criterion; the more general purpose the software, the greater the potential impact.

The analysis so far has ignored effects like network externalities on both supply and demand sides. Software is subject to strong lock-in effects, often driven by user learning: a user develops large amounts of knowledge regarding how to make the software perform the tasks he needs; and through legacy: switching software applications typically demands translation of data files and often communications protocols and settings. This and other characteristics of software use present a large potential for lock-in, which will favour the technology that has first reached a defined installed base.[11]

11 For a formal presentation of network externalities see Farrell and Saloner (1985), Katz and Shapiro (1985). Arthur

This is why, on the supply side, firms are competing strenuously both to have their products become the standard and to obtain a large installed base of users. This is usually done using low penetration prices and announcement of products before they are commercialized. If a user is locked in, the appearance of an entrant (D in Figure 2) does not necessarily help—even though the new product fits better the user's needs, it may be too costly to switch.
The release of OSS code can play into this issue as well. The difficulty of switching can be characterised as an issue of standards. If file formats, communication protocols, user interfaces are standardized, switching costs fall dramatically. If switching costs are low, barriers to entry are low, since it is easier for a new entrant to attract users, and thereby build an installed base (and thus sell more copies of each version of his product). To the extent that released OSS is taken up by different developer firms, this represents at least some degree of standardization, and thereby lowers switching costs, barriers to entry, and can lead to an increase in user utility.
Again, it is relatively easy to see where this effect will be strongest. If the released code is particularly good in terms of user interface; file formats per se; file format translation; or any other feature that contributes to user lock-in, the released OSS can contribute to the development of open standards, which are well-known to mitigate lock-in effects.

1.6 Conditions and effects of publishing software under an OS licence – empirical evidence

1.6.1 Presentation of cases studied

The case studies are intended to examine the economic, legal, and organisational conditions under which public organisations become able to provide own software under the conditions of Open Source Software licenses and best practices. Another focus of the case studies is to find out whether development and publishing of software owned by public organisations under Open Source Software licenses increases and / or improves the exchangeability of data and interoperability of systems. Finally the environment of the public organisations selected for the case studies is examined, i.e. not only their relation to other organisations with which they exchange data and software but also to local and regional support organisations, the FLOSS community, educational institutions which they might use for training courses related to the development and publication of Open Source Software, businesses, and citizens.

The case studies are selected with the following main criteria:
- covering different levels of government
- covering mainly specialised, but not PA-specific software
- different modes of procurement (impact on market)
- different levels of community support (impact on market and society)

The impact of Open Source Software on the transparency of government can better be analysed when applications are being used for government services rather than when activities are mainly general purpose software This means that the software we prefer to focus on is developed directly or indirectly through a public administration body, or has been adapted substantially to specific e-government needs by public bodies or their agents: "specialised purpose" in the table "Types of software applications" below. The reason for this criteria is that very generic purpose software is unlikely to have been substantially developed or adapted by public administrations, and thus may not have a significant market impact. (There are exceptions, as in the case of Extremadura, where a

(1989) for a treatment of lock-in under increasing returns. For an historical account see David (1985).

general purpose Debian GNU/Linux was substantially adapted through integration with specialised tools, e.g. remote access and terminal monitoring tools, to provide a classroom solution that is market competitive and has both PA-specific and private sector uses.) At the other end of the spectrum, software with a specific PA purpose, such as land records management, has almost always been developed mainly for the exclusive use of individual PAs or groups of PAs. While there have been many calls for sharing among PAs (including the 2002 "Pooling Open Source Software" study conducted by members of the contracting team for the European Commission), such sharing as it is between a well defined class of users does not require to be open source. Moreover, as the market for such software is limited, its impact (e.g. on competition) is not widespread. By lowering costs and broadening use of ICT among PAs, release of such PA-specific software as open source nevertheless has an impact (e.g. on citizen services and transparency of government), but these can also be studied without examining specific cases.

In addition to the criteria for the selection of case studies as they are mentioned in the proposal the contractors also consider the context in which the production or release of open source software through public bodies is executed. This includes the political context (e.g. who drives the open source software release) as well as the market context (who develops the software – the PA itself, a subcontractor chosen through public procurement, or a selected subcontractor).
Another selection criteria is the degree to which the open source community is involved in a project driven by a public body. While, for instance, there is no such collaboration at all in the LinEx project in Extremadura, the MMBase case is at current fully driven by the community. Other cases. Such as the London Borough of Camden, show a medium level of collaboration between public bodies and the community. Finally, another selection criteria was the extent to which the software is released under an open source license.
It should be noted that we have not included cases of "failure". Failures are, first, difficult to identify and, second, it is usually very hard to find a person who is willing to talk about a failure in detail. Moreover, from failures one can only learn at which points such a project fails, whereas successful projects typically faced the same or similar situations, but have overcome them. Thus, studying "successful" cases allows us to learn not only what the problems are, but also how they could be solved.

Extremadura

Luis Casas of the Extremadura government explained: "The LinEx distribution of Extremadura was launched in April 2002. In fact it was the last big action of the regional strategy to access the information society, which started in 1998. The first proposal of this distribution was to provide the regional government with software to be used in the schools of the region. The objective was to have 1 PC per 2 students in 2005. When the government looked for reliable or suitable software they realised they could never afford to pay the licence for proprietary software. The regional government decided to launch a distribution, very localised, that means that it's all in Spanish and also that the icons and the names, everything was related to the culture of the region of Extremadura. It was supposed to be only used at the schools".

The GNU/LinEx system was described by the interviewees as very stable, with very few reports of technical problems and extremely low maintenance efforts. The fact that the whole system in the educational sector could be managed centrally also contributed to a high degree of satisfaction with the developed software. The representatives of the provincial government emphasised that GNU/LinEx is not only a technological project. Much more than that they consider GNU/LinEx to be a social, cultural and economic advancement that cannot be reached by proprietary software

because it does not allow a comparable interplay between political administration, development, and users.

London Borough of Camden

The UK Office of the Deputy Prime Minister funded the APLAWS Pathfinder Project to develop web standards and an Open Source Web Content Management System that can be freely adopted by any local authority in the UK[12]. It was in 2001 when "Camden took on the project and piloted it, using it for [its own] website and at the moment also using it for their intranet", according to Alasdair Mangham of the London Borough of Camden. The project was funded until March 2006. The London Borough of Camden did not use any OSS before APLAWS started. OSS only became an issue because there was a need for a CMS that serves the special needs of public administration and a requirement from the Office of the Deputy Prime Minister (ODPM) to distribute the software for low cost.

As one of the strongest advantages of APLAWS, our interlocutors described the ease of use, flexibility and maintainability of the system through in-house capacities. Every user knows that whenever he has a problem with the system or wants a change he can make a call and it gets fixed or changed very soon and that he does not have to wait for an external company or freelancer.

VPRO / MMBASE

MMBase' CEO Jo Lahaye says: "MMBase is originally developed by a public broadcaster called VPRO. VPRO is one of the most creative public broadcasters in the Netherlands and they worked on this project from late 1995 / beginning of 1996 until April 2000 when they decided to make MMBase open source. The reason was that when they started the MMBase project there was nothing on the market that could sufficiently supply the needs of the editors working at the broadcaster, because these editors were used to using technology in a way that they don't want to hear 'no, it's not possible'."

The main advantage of MMBASE as a FLOSS application was seen in its high degree of flexibility, adaptivity, and openness.

NFI

The Dutch Forensic Institute, NFI, decided to develop as much OSS as possible because they felt a need for software that is easy to understand with regard to how it works and what it does, that is reliable and that can be adapted to changing user needs in the field of justice. As Jeroen van den Bos, the head of the IT department at NFI explains, the decision to use as much Open Source Software as possible did not arise as a general strategic decision: "Often the decision that a system can be made Open Source was not made at the start of a project, but came along at some point in time during the project, just when it became obvious that Open Source could help us to solve a problem that we couldn't save with proprietary software. We are always interested in Open Source, but we never actively promote it right from the beginning. A very important factor for the software produced by the NFI is that it provides "proofs of evidence": "For instance, when somebody gets arrested and the police wants to read out his mobile phone, where all kinds of messages and information are stored that might be important for the police to know, then we must have a software that without any doubt can prove that the messges and information that are read out have definitely been on this mobile phone."

An advantage of FLOSS compared to proprietary systems is that it allows to look at how the software actually works - "this is one of the key reasons for us to use Open Source Software", as the

12 http://www.aplaws.org.uk/project/

head of the software engineering group explained. Proprietary software systems appear as a "black box" in this respect: "Proprietary systems can reproduce the results, but you cannot see how the results are generated." It should be noted that access to source code is not an important reason to use FLOSS for most PAs, according to the FLOSSPOLS survey. Arguably, users of FLOSS attracted by the access to source code are also more likely to develop their own software and/or release software they own as FLOSS.
Another advantage of FLOSS for the NFI is that it is easier to keep the produced software systems up to date: "One of our objectives with using Open Source is that others, such as commercial companies or individual developers, share responsibility for enhancing, maintaining, and updating." Using OSS allows the release of results of R&D at NFI much faster than with proprietary systems, as the user and developer community can immediately have a look at the code or test the system and comment back to the NFI. "The wider commercial and non-commercial community can take them (NFI's software systems) up and advance them." Finally, OSS allows easier exchange of data with other institutions.

Beaumont Hospital

The reasons for Beaumont hospital to migrate towards OSS were purely of economic nature, as the head of the IT department, Tony Kenny, pointed out: "We were confronted with a budget cut in 2001, the overall annual IT budget decreased from €1.6 million to €20.000, which means we nearly had no money at all. In this situation we had two options: either doing nothing at all or migrating towards FLOSS. We made a calculation of how much it would cost to pay for the license fees (for existing software) alone in 2001 and found that this was more than € 600.000. The management clearly said that we won't get so much money. So we presented them the alternative to use FLOSS instead of proprietary software, and the management accepted."

The IT department at Beaumont Hospital considers OSS to be *per se* aligned with a massive quality boost. The head of the IT department explained that though the proprietary software tools (Visual Basic, Delphi) that were used before the step towards OSS was made were easier to use and to understand than Java and though it took quite a while until a comparable degree of knowledge under the new Java development environment has been built up in the department, now the main benefit of OSS is that the IT department has an enormous wisdom available to it that can be exploited (portals, identity management, etc.).

1.6.2 Findings from the survey

There are two main reasons for local governments to release software under an Open Source license, i.e. of those who either have already released or plan to release Open Source software in future, which applies to ten per cent of the respondents. One relates to the Open Source community and philosophy and thus to be considered as an external motivator ("to contribute to Open Source (freedom of choice regarding software)") and the other one relates to service quality and thus an internal motivator ("to enhance the quality of the services provided by organisations like mine"). Apart from these two motivations, vendor independence and better customised software are also needs that play a vital role for public bodies to publish their own software as OSS. Interest in the economic situation of the region in which the public body operates however plays almost no role at all as a motivator (see Figure 3).

The survey results imply that the willingness of public bodies to publish own software as Open Source software might be as much dependent on the services that are provided by this public body as on the public body's involvement with or understanding of the Open Source philosophy. This

opens a wide space for political initiatives aiming at increasing public bodies' contributions to Open Source software: They can, on the one hand, focus on the services provided by the public bodies and develop strategies that help implementing these services in software systems. But on the other hand they might already achieve a great deal by just informing public bodies about the idea of open source and by arranging contacts between Open Source actors and the public sector.

Figure 3: Reasons for public bodies to release software under an OSS license – survey results

This observation is fully in line with our findings from the case studies: They all were familiar with Open Source software, its ideas and principles. This familiarity was usually provided by individuals in the organisation who, in a certain context of decision-making on the implementation of services into software, were able to convince the organisation as a whole to decide for Open Source.

1.6.3 Organisational issues

When deploying Open Source software (or any new software, for that matter) in an organisation, good change management is key to a smooth transition. The development and publication of Open Source software bring a different set of challenges, which may be new to the public body in question. In order to turn the software they released into a successful project, they may have to monitor changes made by others, control the quality of the software and take into account the needs of users.

1.6.3.1 Sound management, e.g. through a technical committee

The purpose here is to create and manage a technical committee that will take decision to develop or not new innovations that will be implemented in the next version of the product based on the user needs. "User needs must be taken firstly in consideration while developing a product" (MMBASE): this is a directive that can be drawn as a conclusion from all our case studies. The technical committee must be present to ensure that the user needs coming from different sources (forums,

user groups, etc.) can be translated into a technical requirement. This need implies that this user' request can be processed and that a technical tool can support it and track it.

In addition, the public body that publishes OSS usually has an interest to ensure that no organisation will fork the application and create discrepancy between two products having the same name. The “no forking”-needs presented by MMBase and APLAWS was a guarantee to stay the “owners” of the base code and ensure the viability of the project and the viability of the support that can be provided to the users' base. This vow couldn't be avoided in any manner, neither intellectually nor technically due to the OS licences. Until now, neither MMBase nor APLAWS have been forked.

1.6.3.2 Maintaining quality, including security, user-friendliness and documentation

The purpose here is to keep a good quality product in term of performance, security, user friendliness, and documentation. Main technical issues have been resolved by our by the public bodies we examined in our case studies by implementing a state-of-the-art development model. They know the difficulties to reach a certain level of quality in each of those criterion's, and therefore we often found documentation as their main concern.

In order to distribute an OSS a lot of technical procedures have been put in place by our case study subjects:

- Writing and executing test cases to ensure that any upgrade, proposed by an internal or an external contributor, has no impact on the product behaviour. This is part of the reviewing process.
- Performing performance testing to ensure that any upgrade, proposed by an internal or an external contributor, has no impact on the product performance
- Usability testing of the products to ensure that any upgrade still keep the user interface of the product easily usable by a standard user
- Writing and correcting the documentation.

In general, the projects we examined have put in place the tools that allow them to produce a state-of-the-art OSS. By default they had to set up a technical committee (see above) but also to set up all the tools to support the development like Version Control System, automated software build, newsletters, support website, etc.

1.6.3.3 Programming and maintenance capacities (“readiness”)

A crucial factor in this regard is the level of development that is done by the organisations when sharing their OS developments (sorted from the lesser to the heaviest development work). Lack of prior experience with OSS is likely to be a barrier to using – and releasing – OSS software. The degree of OSS usage before the public bodies' OSS projects is extremely different within the five case studies:

The provincial government of Extremadura did not have the software development capacities that were necessary to develop and maintain the required software systems for the educational sector in-house. The provincial government did not use much software in general and no OSS at all before LinEx started, because the public sector and in particular the educational sector did not have many computers.

Representatives of the provincial government consider themselves to be “public service providers”,

not software developers. GNU/LinEx provides an example of an “externalised customisation” of an existing software product. Extremadura customised lightly the user interface of the products they use by changing the name of the application based on, for example, famous Spanish citizens, artists, etc.

On the other hand, NFI, Camden, MMBase, and Beaumont Hospital develop brand new products which imply some huge technical impact depending on their user needs and their wishes to keep an eye on the external development based on their products. The London Borough of Camden had a team that was capable of designing, implementing, and carrying out a software development project, so the organisation was ready when the tender from the ODPM was published. We call this type of readiness “experience-based innovation. This type applies also to the cases of VPRO and NFI.

A third type of programming and maintenance capacities for public sector OSS activities – besides Extremadura's “externalised customisation” and London Borough of Camden's / NFI's /VPRO's “experience-based innovation” - is provided by Beaumont Hospital. Beaumont Hospital had no OSS experience at all before they decided to migrate to Open Source. They actually had a Linux server, but nobody was aware that it was Open Source and how the server worked. The reason to migrate towards OSS was severe cost pressure, and only after the IT department had made good experiences with OSS after this decision they began to extend their OSS activities and to promote their products. Therefore, to develop software was aligned with massive organisational and technical barriers because nobody in the IT department had any knowledge of FLOSS, no FLOSS systems were used by 2001, the staff had very limited systems management and systems engineering skills, and many people only understood GUIs. We call this type “cold adoption”.

1.6.4 Economic and financial issues

The following section relates benefits for public administrations releasing their own software to concrete findings from the case studies conducted for this project. It should be noted that reported benefits are often more closely tied to the *use* of FLOSS rather than to the release of code as FLOSS. The two are in fact very closely related – the release of software as FLOSS is often prompted by having used FLOSS as a basis for software. Licensing conditions can sometimes require the resulting software to be released as FLOSS, and even when this is not the case, users of FLOSS learn about the FLOSS community and are inclined to release the software they develop as FLOSS.

Extremadura

Since there was not much proprietary software used at the Provincial Government before GNU / LinEx, the real cost and savings cannot be calculated exactly. Relatively exact figures for some components exist only for the first year (2002). Asked for an overall balance, we were told that the Government spends the same amount of money for IT as before GNU / LinEx started, but gets more for the money in terms of quantity and quality. It should be noted, however, that the motives of the Extremadura government went far beyond economic questions. The aim of the project is not to save money, but to promote digital literacy in a traditionally poor region.

Money which would otherwise have been spent on software licences could be saved and instead be invested in hardware (infrastructure). There is no comprehensive and exact calculation of the monetary benefits of GNU / LinEx, but as a lump sum for savings of software license fees the Provincial Government calculated a total of 30,000,000 €. Some additional money was saved

because of re-usage of code that was developed in province of Andalusia, but this position could not be quantified.

London Borough of Camden

The costs aligned with APLAWS were hard to quantify for our interviewees. It was evident for them that the effort resulted in a very positive outcome, and since the Borough got funded by the LAWs National Project costs were not paid from its own budget. The local authority has a support contract with Red Hat ("usual terms and conditions") that is reviewed and – after positive review - renewed annually. Apart from that, London Borough of Camden pays a web-designer to provide the look and feel of the front end of APLAWS, and it also pays for the quality assurance.

Although the APLAWS administrators could not provide detailed figures on the cost and benefits of the project, a document provided by the LAWs National Project administrators and the ODPM indicated that APLAWS (in combination with the LGCL) provided a significant monetary benefit, as shown in Table 1.

Per LA Annual Financial Efficiency Savings from Implementing LAWs Products[2]			
	Low (£k)	Average (£k)	High (£k)
Per Authority:			
County	13	21	28
Unitary	12	19	26
District	10	15	20

Table 1 *Source: ODPM (December 2004:) The benefits of LAWs.*

Monetary benefits in terms of low cost for software licenses and training, as training costs could be reduced because Intranet and Internet use the same system. As a consequence, staff time for training could be reduced through reducing the number of training courses. The same positive effect is achieved by sharing documents.

MMBASE

As with APLAWS, cost aligned with developing MMBASE could not be calculated exactly. One reason for this is that the development of MMBASE started already 11 years ago but became OSS only in 2001 (whereas the other projects we examined started all around or after 2000). Another reasons, which is however common to all projects, is that many cost are not considered to be a unique issue of OSS and that some of the cost cannot be calculated because they simply do not exist. For instance, personnel cost are the same if a developer is employed in-house, regardless of which kind of software he is producing. Cost that do not exist are : "Original costs? How are you going to calculate on 15 people developing for 5 years on a project? I mean before it was OS? (...) I don't think they (VPRO) would like them (cost figures) to be out there. Because of course, this is not a very cheap project. You can imagine. And it's not like all these 15 people didn't do anything else then just concentrate on MMBASE."

NFI – Dutch Forensic Institute

The head of the software engineering group of the Digital Technology Department at NFI confirmed that using OSS systems saves a lot of money in terms of license fees, but with regard to investments in R&D he sees no differences to using proprietary tools. The “basically only significant cost” he sees in the R&D context is the cost for a man-year of work, and that, he says, cost the same in OSS as in proprietary software. He admits however that R&D is eased and accelerated with OSS. He didn't want to specify how much NFI saves through OSS usage because this point is not relevant for NFI – they do not use OSS primarily because of cost savings, but because of other advantages such as transparency, reliability, or ease of maintenance.

Beaumont Hospital

With regard to budget, the head of the IT department at Beaumont said that budgetary arguments are often a too simple answer to the question why OSS is applied, but on the other hand he presented the fact that in the strategic plan for 2002 to 2004 it was estimated that implementing 26 systems would cost something about € 4.2 millions, but now it turned out that 19 out of these 26 systems have already been implemented and that this didn't consume more than about € 700,000 (these € 700,000 were spent for everything, not only Open Source, approximately 50% of this sum were spent for OSS). “So we made significant progress, and I think we wouldn't have realised this if we wouldn't have utilised Open Source.” But there's no comparison of cost using proprietary systems versus costs using Open Source systems.

1.6.4.1 Spill-over effects

Most public bodies we interviewed – except for NFI - said that there are significant spill-over effects of public bodies' OSS activities. The strongest impact of OSS activities on the scope of OSS usage in public administration could be observed in Extremadura. At the begin Extremadura didn't know the impact they will have by setting-up a Linux distribution. “We started at the schools and we made a distribution to the local newspapers of the GnuLinEx CD” but quickly the “...civil society and also enterprises wanted to work with [the GnuLinEx distribution]...”; clearly the scope of Open Source Software that was used in Extremadura changes after the decision was made to develop and distribute their Linux distribution.

The wider impact that OSS had in the information society in Extremadura led, in turn, to changes in the use of FLOSS within the government. Originally limited to education, FLOSS solutions were adopted for healthcare and hospital management, and in 2006 a decision was made to require all public administration use of software to be FLOSS, including for ordinary government activities.

Since there are no figures about licence sales or similar data, it is often difficult for public authorities releasing Open Source software to quantify its use outside of the releasing body. Download numbers provide only a weak indication. Reports about users who adopted the software, often other public authorities, are more helpful. Similarly, the existence of consultants and developers for the software indicates that there is a market for their services.

In our survey, 48% of “releasers” assume there would be a positive economic impact if local governments would release their own software under an Open Source license. They think that then local enterprises could provide support, training etc. and create new employments. What characterises the expectations of the “non-releasers” is that they assume there would be no

economic impact at all (53 per cent). Neither the “releasers” nor the “non-releasers” expect a negative economic impact of public bodies releasing their own software as Open Source, and that there is some confidence that it might have some positive economic impact, if any.

1.6.5 Reliability, extensibility and security issues

In the case studies, none of the public bodies we investigated saw the openness of the code as a security risk. According to them, users and citizens are more concerned with the confidentiality of their personal data in the system, independently of the software or its licence.
Rather than the quality and security of the code, interviewees reported that it is a constant challenge to obtain and maintain good documentation for the software. Also of concern was that contributions to each project, especially those from the outside, should follow determined coding conventions, so as to be more easily integrated.
Our survey results show that extensibility and reliability play an important role when public bodies consider releasing own software as Open Source, but security concerns are less pronounced and appear definitely not as a barrier for public bodies towards releasing software under an Open Source license. This is completely in line with what our interviewees told us in the case studies interviews. Fears of cost associated with reliability, extensibility, or security do not deter “non-releasers” from developing and distributing their own software as Open Source, and neither do security concerns, though the latter are clearly more pronounced among the “non-releasers” than among the “releasers”.

1.6.6 Technical issues

OSS development is characterised by network-based source code revision, code testing, bug tracking, communication with internal and external developers, communication with internal and external users, software build management, and performance profiling.
The public bodies that we studied solve these tasks in much the same way that other free software projects do. Review of the source code is handled either in-house or contracted out. For bug tracking, they use the facilities provided by forums such as SourceForge. Communication with internal and external developers happens by a variety of means: instant messenger tools, website, newsletters, postal letters, fax, committees, etc; it should be noted that face-to-face interaction has an important role for the integration of user feedback. MMBase and APLAWS put emphasis on ensuring that contributions follow their coding conventions; a modular design approach is also important. Providing an efficient support is a major concern for all the interviewees. They have set up support in different ways: on-line support through a website, mailing lists and support from third parties (training company or IT service of a company using their product).

The cases studied solved different technical problems solved through OSS. Extremadura had a lack of technical infrastructure to address. Both the APLAWS and MMBase projects serve the need for CMS with special capabilities. The NFI develops OSS to alleviate a lack of understandable, transparent, and reliable software applications that can easily be advanced or adapted to changing user demands in the field of criminal justice.

The degree to which the local software and hardware infrastructure was affected by the use and development of free software varied considerably. The most fundamental change of software configuration and hardware architecture could be observed in Extremadura, where the number of PCs in the educational sector reached a very high level (e.g. 1 PC per 2 students in schools) that would not have been reached in such a short time with proprietary software and where completely new IT networks and communication facilities (website) had to be built up.

Beaumont Hospital also experienced a high degree of changes in their software configuration and hardware architecture because the IT department had to replace almost all proprietary software APLAWS and MMBase both use the SourceForge infrastructure to provide facilities for developers and users. The lowest degree of change of software configuration and hardware architecture has been observed at NFI.

1.6.7 Legal and regulatory issues

For public bodies developing OSS, legal concerns mostly refer to the choice of licence for the project. As will be discussed in chapter 3, procurement or competition rules rarely pose an obstacle to the release of public sector software.

Our interviews demonstrate that the GPL licence is well known, and the fact there are many open source licence models is known too. A licence is generally chosen after consultation of internal staff (we found no case where external specialised lawyers were involved at the beginning of a project) and in this case, the choice is often made by developers (and not by lawyers). The choice of licence may have a considerable impact on the project's chance of attracting a developer community. Best practice is to dedicate at an early stage enough time to select the licence, and to share the decision between legal experts (external if no internal expert exists) *and* those responsible for software development (the latter know better than lawyers the impact a choice may have on the growth of a supporting community).

The government of Extremadura prefers to use the GPL whenever possible, as does Beaumont Hospital. APLAWS is distributed under GPL and LGPL. As the project has grown, many different organisations and individuals have contributed to APLAWS, each retaining the copyright to “their” parts of the code. Thanks to the GPL, this is not a problem; as soon as each developer distributes her code, it becomes available to all others under the conditions of that licence. The MMBase administrators released their project under the Mozilla Public Licence (MPL), a weak copyleft licence, in the hope of achieving faster uptake than under the GPL. In fact, the opposite happened: Businesses were reluctant to develop the software further, because they were afraid that competitors might make derivative works of their products and release them as proprietary software (something the MPL permits, but the GPL does not). The NFI releases its software under a BSD licence, as it is of the view that different from the GPL, this does not exclude commercial partners.

Our survey results state that only one third (36 per cent) of the governments owning software publishable under an Open Source license claimed to have full rights on this software. Also, a clear majority of those who contracted the software development out said that the contractor provided them with full rights on the software, but overall the number of cases is too low to provide statistically reliable results for this group.
Only 3 per cent of the local governments that participated in the survey have already released software under an Open Source license. Interestingly, three of these seven public bodies belonged to the group of respondents that do not currently own software that can be published under an Open Source license.

1.6.8 Effects on government services

All case study interviewees emphasised the importance of interoperability, but they did not see this issue related to using or not using OSS. The most important aspect regarding interoperability was clearly seen in using open standards or not. Several European countries provide interoperability

frameworks and push the usage of OSS but, they don't talk about the contribution to the OS Software.
In the OSOSS surveys as well as in the FLOSSPOLS local governments survey interoperability turned out to be one of the most important reasons for public administrations to consider Open Source Software. Improved exchangeability of data between departments as well as between different organisations was considered to be achievable through the use of Open Source Software. Although open source and interoperability are not the same thing, they are clearly seen by public sector users as being related in that interoperability, open standards and the resulting vendor independence are perceived to be associated more with open source software than proprietary software.
The impact of developing and using OSS in public bodies on the transparency of their services is evidently quite limited. “Usage of Open Source or copyrighted [proprietary] software has no impact on the transparency of the government activities” and “citizens are only concerned by the security of their transactions independently of any software considerations” says Extremadura in strong concordance with all other public bodies that were examined for the study. Most PAs believe that interoperability relates to open standards and while this may be supported by (and should support) FLOSS, it is an independent issue.

2 Report on possible support for the deployment of OSS

Our purpose in this chapter is to address the dynamics and mechanics of releasing software fully owned by public organisations under open source licenses. We will discuss the following aspects:

- the role and/or need for OSS competence centres to inform and guide potential users throughout Europe on the availability of particular open source software and on best practices;
- the specific technical requirements (such as availability of specifications, code documentation, language considerations, business process descriptions etc) related to the software to enable efficient re-use and sharing
- Is an open source community required for the development and maintenance of software used by public organisations? Is it realistic that such communities would emerge and be self-organized, or what other models for such communities could apply?
- What could be the role of the software and services industry?
- Are there existing repositories suited to this task?

2.1 Aspects of a successful Open Source Support strategy

We will first examine the global aspects of implementing an open source strategy. This question may be independent from the fact the software is "produced and distributed by the administration" or if it is not, ore only for a part. We see globally four main steps that may also be seen as four phase of an iterative loop (after final evaluation of a cycle, initiate the next one):

- Motivation / Knowledge sharing
- Assessment (*ex ante* evaluation*)*
- strategy definition
- strategy execution

2.1.1 Motivation / knowledge sharing

The first step should be dedicated to knowledge sharing, motivation, understanding the opportunities, but also the risks and possible issues related to an OSS approach. It is difficult to assess how far this was done inside the "historic" European open source case studies, as the Extremadura region, Camden etc. Most pioneer cases are generated by early advocates, with enough persuasion power to initiate a dynamic. However in all teams there are leaders and followers, there are employees that are now using at home a stack of software, usually proprietary software that they "did not had the impression to pay for" just because it was included in the purchase of their new PC. These employees could seen as a necessity the idea to migrate to other tools and change habits. In large Public Administration where the management is not advocating a specific business model, but is rather sceptic about too beautiful promises and rather careful to the idea to change dramatically the way the Public Sector is working, such awareness rising exercise will be very useful.

This contributes to the construction of a vision: regarding the distribution of public sector software, the first support has to be built internally. The public sector staff will be the first ones to animate a

supporting “community” and should therefore have the feeling that they have now an opportunity to “move the change with autonomy” rather than just applying the decisions taken by the proprietary software industry on what is convenient or not.

2.1.2 Assessment

The project to implement an open source solution must be initiated with an assessment of products or “components” that could help the development team to start the project based on an appropriate architecture. A preliminary question is “does the solution exist somewhere?” Is it available, possibly without licence fees? Under a convenient OSS licence? Could we contact the developers community? If no similar solution exists, could we at least use existing building blocks? Almost no solution is developed totally from scratch today. Most are depending on the acquis of ready to use library or environments.

After determining building blocks, the impact of integrating all components into a solution and proposing it to “a market” must be evaluated. Using a proof of concept of the proposed software, it should be evaluated whether the intended addressees of the program are interested in the solution; if the organisation is ready to enter into the long-term commitment of maintaining a Free Software project; and if it is able to handle the technical complexities.

2.1.3 Strategy definition

Once the proof of concept established, the medium and long term strategy can be defined based on the various options, and by selecting the most appropriate. If necessary, an outside consultant can be asked to estimate the cost of the project, clarify legal issues, optimise the development structure and determine the appropriate open standards to use.

2.1.4 Strategy execution and evaluation

Depending on the case, the execution of the strategy and the related “change management” will be more or less complex. In particular, the possible decision to develop its own software or to distribute it under an open source licence is only one of the aspects on public sector ICT management. It could and should stay a “minor” aspect in the sense that the core mission is different: it is the quality, reliability and continuity of a service to be delivered to citizens and enterprises. Technical evolution is just a means to achieve the objective. The desire to be “state of the art”, “first ranked in eGovernment applications” sometimes has to be moderated by the weight of legacy documents (migrating data is always difficult, in the direction of open source or not) and integration in a complex, mixed set of other applications.

2.2 The role of and need for OSS competence centres

Facing the complexity of the strategy, the variety of the case, the diversity of the solutions, and the issue related to software and knowledge pooling, there is a need for support from specialised centres in a variety of domains.

2.2.1 Licence questions

There are today more than 100 different licence models. Many of them are not compatible each other. Therefore it may become difficult to create a solution build on different open source components and to distribute it under a single licence.

Most licensing models are designed by US lawyers for US users. Certain specific European Copyright concepts are ignored. The provisions regarding warranty, liability and consumer information in general are ignoring European directives and national laws, making some of them at least questionable, if not totally invalid. The provisions regarding applicable law (US, California etc.) and competent court are generally not acceptable for European public sector either. Last but not least, the licence language is English only and they may have no legal value in other languages, even when a translation is available "for information". Providing the sole official value to an English text to manage relationships between – for example – the French administration and a French citizen or enterprise, looks strange and could also be simply contrary to applicable laws.

The role of OSS competence centre is to document such issue and to explain how to solve them, by selecting compatible component, by using convenient contractual models with contractors and developers. Provided it will be – on its turn – an OSI compatible licence, the European Union Public Licence (EUPL) is compatible with European concepts, warranties, liabilities, applicable law and competent court. It could solve several of the other licensing issues, under several conditions:

- The list of compatible licenses should include most OSI copyleft licenses (a condition is to obtain the reciprocity from the relevant organisations). This could allow developers
  - to improve a EUPL licensed work with any other copyleft component and to redistribute with the relevant other copyleft licence;
  - to include the EUPL work into another work and to redistribute the other work under its original copyleft licence
  - (if reciprocity is obtained) to solve incompatibilities when integrating several non−compatible copyleft components into a EUPL licensed work.
- Several linguistic versions could receive the same legal value, at the condition to be certified, e.g. by competent legal advisers.

2.2.2 Statutory questions

Mission

The fact distributing own software as Open source is not neutral and has an impact on the political image of the organisation. Is it compatible with the "statute" of such organisation? The first question is related to the "mission": there are no doubt about the necessity to obtain specific public sector software, at the best possible conditions (best value for money), but do this imply a "redistribution activity" as open source provider? The answer to this question is that – if Open Source development provides really – by the mobilisation of volunteers and the creation of a community - the best value for money, the publication of the source code is a condition for the creation of such community. If best value is obtained by sharing resources (e.g. by associating efforts from several public sector bodies, possibly from several countries) to produce common and interoperable software (e.g. in the field of eProcurement) then sharing of this source code is a pre-condition. If not ensured directly by public sector, redistribution and the control of collaborative development could be ensured by the OSS competence centre (that may be external to the public

administration, for example a national or European non-profit NGO).

An alternative to re-distribution is to use open components to develop internally (with officials or contractors) and support all costs. Indeed, redistribution is not an obligation. This is the way chosen by some Ministry of defence developers (in France, in US) for critical application. However, there is – in such case – no sharing of new developments cost and reduced external impact.

Ownership

The question of Ownership is another statutory element. European Commission statutes (and national rules) specify that what is realised by the public administration belongs to it "exclusively". Similarly, if a software is developed based on public funding (taxpayer money) how could it be "given" to other public authorities (from other countries) or stakeholders? Interview demonstrate rational and pragmatic attitudes facing such question: the relevant public bodies know that by licensing their software under an open source licence, they do not "give away" any parcel of their IPR. They keep full ownership and authorship. At the contrary, copyleft licensing opens for them the possibility to obtain improvements without having to pay for it.

2.2.3 Procurement questions

A third question is related to procurement: Producing or purchasing public sector software is often done via public call for tender. How far is the purpose to be authorised to redistribute the purchased software as open source compatible to a fair evaluation of received tenders?

If the software is very specific (e.g. to be developed specifically to address public administration needs), we do not see any problem here, because nothing in this requirement may damage the business model of potential contractors. However when the software appears as a commodity (general purpose software for office, calculation, presentation, statistics, communication etc.) a global redistribution right is the "negation" of the business model of some contractors. Open source advocates says that proprietary software industry is "free" to adopt open source licensing, but in fact these are not free and their managers could have serious problems in doing that (they would lose their jobs because of legacy, investments, shareholders, stock market and asset management rules etc.). They cannot change that business model easily by authorising "world wide" gratis re-distribution of their proprietary database or office software. In such cases, the requirement concerning predefined "Open standards" looks more appropriate and comparison between tenders could be based on obtaining best value for money. Fair TCO comparison between OSS and proprietary software are sometimes extremely difficult to proceed, because of the multiplicity of case-specific cost factors, including human costs, migrations, retirement etc. and because the level of guarantee and services obtained both sides is not easily comparable. These commodity software are generally out of the scope of specific Public sector software and therefore the above aspect of procurement does not represent a serious issue in our case.

2.2.4 Providing Internet tools and collaboration services

Building critical mass

For the deployment of a effective pooling and support service, it will be quite important to gain as quickly as possible the critical mass which converts a good design into a productive and useful site. This means:

- Early public availability of all design documents not considered as confidential by the public administrations or the contractors, and dissemination of them not only in a public website,

but also in conferences and academic or journalistic publications.

- Contacts with other OSS projects, public administrations subcontracting OSS development or developing it themselves, and companies interested in the deployment of OSS in public administrations, with the aim of both getting feedback on the design and attracting their interest in the project. This is one of the opportunities presented by the new European open source observatory and repository ("OSOR") project. This will foster the migration of development projects and speed up the reunion of a critical mass as soon the initiative will be in operation.
- Dissemination within the OSS community, including the meetings of most important OSS projects and OSS developers in Europe (FOSDEM, GUADEC, EuroKDE, ApacheEurope, etc.) - Agreements with already existing similar sites, either to group part of their activity to a common service, or to define federation collaborations. The sophisticated benchmarking and multilingual features of the collaboration tool are seen to be leading attractors for such collaboration.
- Quick population of relevant information publicly available in the Net, so that casual visitors may appreciate the value of the site for them, and decide to come back. Today´s visitors will be the core of tomorrow contributors (as OSS development has taught).

Developing European synergies with involved organisations
The implementation of an efficient support service if far to be a simple technical work: it is developing leadership to motivate permanently OSS communities to contribute, provide them incentives and advocating the Public Sector representatives to progress in pooling their software under Open Source license.
Based on the assumption that open source developers' communities and the projects they develop tend to be driven by self-interest, the motivation for "collaboration between mutually complementary projects" will be based on the opportunity to increase benefit of software distribution: the more open source software is distributed and effectively used, the more it will obtain perennial character and support. Therefore European distribution and visibility (as planned by the "OSOR" are by nature more interesting than local ones.
Stakeholders willing to contribute to the OSOR must share this point of view. To this end we have to make them perceive the OSOR "as their own" service. This vision is depending on the "consensus on the principles" and on a management of the OSOR that should be open to the stakeholders.

2.2.5 Project/product issues

We consider three types of development to which most Free Software projects may be assigned, according to the circumstances of their development: Semi-commercial (i.e. with external funding); in-house development; or a fully community-driven project.

It is essential to clearly communicate the mission of the project, as well as the procedures for decision-making and a route for contributors' advance towards greater responsibilities according to their merits. Clearly defined coding conventions are also essential.

The software itself should be highly modular and permit small contributions to be made, instead of consisting of monolithic code. Quality assurance could be provided e.g. through software certification by a specialised office.

2.2.6 Existing repositories and competence centres

There are many cases of open source repositories, of different kinds. Some provide collaborative development environments (CDEs) and forums for community interaction, others only provide software for download. Some examples are ADULLACT[13], the FSF Free Software Directory[14], and SourceForge[15].

Not all of these are sites hosting development projects, and not all of them are hosting project developed by or with public sector funds, however the simple fact so many centres exists seems to demonstrate the need. The fragmentation of efforts is – at European level, a serious issue. As the American "Source Forge" success story demonstrate it, the interest of such support site is directly proportional to the number of supported project, and the motivation of public sector to participate to a common (multi-lingual) repository is one of the most challenging issues at European level.

We have here to expect the result of the OSOR feasibility study, that will analyse several aspects of each site, including at least (and according to the availability of information) the hardware and software architecture, the main parameters of performance (number of users, of hosted projects, of open accounts, of page downloads, traffic, etc.), the funding model, the staff, the services provided and the operational costs. One of the result of this analysis will be a comparative description of each significant repository site, noting its organisational, legal, technical and social (from the developer community perspective) relevance to a prospective common (European) repository, including its justification, based on the proportionality and subsidiarity principles.

2.3 The need for an OSS community

One of the main purposes of releasing software as open source software, other than contributing to the general public good, is to reduce the costs of maintenance and further development by sharing responsibilities with others. This is, by definition, dependence on a "community", although this may not always be what is commonly known as the "open source community", rather, it could be a subset. But in fact, even well known open source projects do not rely on a generic "open source community" but on their own communities, and this is tied to features of the development model of open source that are important to understand.

The development of software and the cooperation in open source software developer communities differs considerably from developing proprietary software in firms. Open source software is developed as a result of free cooperation of autonomous developers connected through a highly volatile network organization; proprietary software is usually a result of hierarchical teams working on clearly sequenced tasks. However, open source software, like proprietary software, requires developers who write code, coordinate work, and maintain and administer the progress of a project. When an open source software project reaches a critical size, individual people take response for different project tasks and coordinating institutions and governing structures emerge. These differ from project to project based on various criteria, including the project's ties to firms, size, and technical nature, and can include "benevolent dictators" (e.g. Linus Torvalds as leader of the Linux kernel development), "rotating dictatorship" (e.g. the Perl scripting language) and democratically elected committees (e.g. FreeBSD, Apache, Debian). Technical merits, effort contributed and reputation play a significant role in the determination of leaders and leadership structures. Based on studies of demographics it appears that open source software developer communities contain not

13 http://adullact.net
14 http://directory.fsf.org
15 http://sourceforge.net/

only technical expertise but also a high degree of management skills.

2.3.1 Developer's motives

There is a large variety of assumptions as to why people join the FLOSS community and "work for free". The first thing to realise is that developers, by and large, do *not* "work for free". They perceive themselves as selfish and self-interested, and expect returns from their contribution. These returns are not always monetary: learning new skills is the most commonly cited reason for developer participation. But monetary returns are an important benefit – roughly 50% of developers earn an income through their open source software work, and this accounts for a large majority of the most productive and experienced developers.

Recognition by peers and society is a motivating factor, but the nature of reputation is such that it can reward a few leaders, but not the majority of voluntary participants. It is, however, a factor that can be artificially boosted by PAs at relatively low cost, by awarding prizes or other forms of public recognition to developers.

Another class of motivators is product-related: developers who have a need that an existing open source software product can't meet may be willing to fix the software and contribute their fix back to the community. Similarly, the open source software community is excellent at spotting and fixing bugs in their software.

One key underlying motivator, that often ties into those outlined above, is an overlap between individuals as developers and the same individuals as users. Generally speaking, the pool of developers who may find some self-interested motivation to contribute to a software project is proportional in size to the project's potential user base. This can be disadvantageous for PAs when they wish to build PA-specific software (and advantageous when they want generic software tailored to their needs, as with Extremadura's GNU/LinEx based on the generic Debian package). However, the open source software ecology has many examples of niche communities of users with their own niche community of developers, from precision engineering to hospital information systems. This attribute can benefit PAs; it could lead to many developers from many PAs and PA-related businesses collaborating in a PA-specialised niche of developers.

PAs need to understand in depth what motivates developers, and before attempting to create community-based open source software applications will need to have a clear understanding of why developers would be interested in collaborating. Providing direct financial incentives by paying a developer (or a group) for a certain task is one possibility. Another one is a "bounty" system, with monetary rewards for developers who solve particular problems.

It is important to clearly grasp that there is no single monolithic "Open source software community", but rather lots of different overlapping communities and sub-communities addressing needs of different groups. These communities tend to automatically be close to the users whose needs they address as they often are formed out of those users, or subsets of the more technically adept users. So, open source software in healthcare is largely supported by IT staff at hospitals and healthcare providers along with businesses and supporting organisations around them; open source software for bio-tech (e.g. BioPerl, the basis for the Human Genome Project) originated at the European institutions contributing to the HGP. Similarly, open source software for public administrations is likely to be supported by a community initiated by PAs themselves.

This community may include a handful of independent, volunteer developers – but is more likely to include developers employed at PAs, employees at firms that work (developing software or providing ICT services) for PAs, or freelance contractors who get paid by PAs or supporting firms.

While such communities occasionally arise spontaneously, and are often self-organising once they start, they often need encouragement at inception. This is especially the case when the software at the core of the community is perceived to be "owned" by a single organisation, such as a specific PA.

2.4 Possible support roles for the software industry

The first opportunity for software industry is to participate as contractor (e.g. fixed or T&M price development contract to provide Public sector software) or as resource provider (e.g. Framework contract where industry transfer resources to the public administration, based on daily price per profile).

Another important business opportunity is to act as a unified point of contact. With time, in nearly all type of organizations, ICT managers or simply end users have (in a more or less planned way) downloaded, tested and integrated specific open source software components in their ICT systems, and may be planning some open source migration or developments. At the same time, certain organisation have developed (or purchased) specific software that may be or not – from their point of view - "open source" in so far they have full access to the code with right to modify it and – possibly – to redistribute it.

Public sector is no exception, with additional characteristics : facing a mixed environment made of specific legacy software, new specific software developed internally or developed by contractors for the public administration, most of the time combined with proprietary software applications or platforms. In addition, open source software applications have been downloaded by users, by ICT staff or by the ICT management, and some of these Open source components are already or should become essential for operating the "technical core" of the organization. An appropriate level of maintenance is therefore necessary regarding this "mixed stack".

Open source software support is – in general - difficult to obtain globally, because the characteristic of free/open source software is a great fragmentation of "providers", which are often unstructured communities. Real Open Source enterprises exists (a typical case is represented by the European – Swedish MySQL database provider which was from its start in 1995 set up as an economic enterprise). Quite often, public administration select open source software from these economic actors rather than from informal communities, because "enterprises" appear more trustworthy to cover all possible issues (this is the main reason why the European Commission data centre uses Red Hat Linux for example, and not Debian). These strong economic actors are still the exception and such enterprises have generally reduced local support capacity: this support is entirely dedicated to their product and does not necessarily cover integration with other products, including proprietary ones. Lack of organised support is obviously even higher if software was originally produced by another public sector body (possibly established in another country): the mission of this public body is to provide government services to their own citizens, and not to support other (possibly foreign) administrations.

The risk for the public sector "chief information officer" integrating open source software in its critical applicative stack is therefore to be "left alone" facing possible issues. "By nature" risks are not higher than with proprietary software, but they exists as well. As there is no "global vendor" covering integration (even if we expect that OSS use open standards, no integration with other components is granted or realized), there is a need for an experienced integrator that will propose a solid Service Level Agreement.

Open Source support should be designed as an one stop shop (preferably a single point of contact)

allowing users to receive answers and solutions covering not only software « bugs » but also a wider panel of services related to the integration of the OSS in a mixed environment and to other support and consultancy.

Support that is not related to a specific software stack might include an assessment of the situation and the discussion of strategic questions. It may answer legal and financial questions, or evaluate software that has been developed. It could also help the requesting authority to find an adequate software solution or find the most promising development path.

Support related to a specific software stack may include advice on handling things such as incident processing, roll-out, change management, technological surveys, knowledge sharing and reporting.

3 Guidelines and practical tool kit for making publicly owned software available under an open source licence

This section investigates on the Guidelines and a practical step by step approach for the publication of software owned by public organisations. The scope of Public Sector Software and the reasons of the pooling are first reminded, with reference to the latest declaration of European ministers. Then the development processes are investigated, as the role of communities, associations and other organisations or environment like cooperative frameworks, and in particular the question of the mutualisation (sharing resources) Legal aspects are not forgotten with guidelines and a practical tool kit for the publication of software owned by public organisations covering among others legal, license, and organisational issues. The perspective of an European Public Licence (EUPL) is evaluated. The conditions to check prior to publish software are summarised in a check-list, including quality questions. Finally, a list of common issues regarding the distribution of public sector software are evaluated, preparing stakeholder to respond to the most current "FAQs".

3.1 Scope of "Public sector software"

We understand that "Public Sector Software" (PSS), is not simply any piece of software which a public authority has produced or for the use of which it has acquired a licence. ***PSS is software that translates the law (broadly understood) into operational information processing inside the public administration or between the administration, enterprises and citizens***. Law, in this broad understanding, includes the EU treaties, national constitutions, national laws and derived regulations, but also some case law, circulars, guidelines or other government instrument defining typical internal or external government processes.

With such definition, the border between PSS and other software seems easier to trace: a solution allowing taxpayers to declare income, an application designed to manage public infrastructure, land, public roads or cemeteries, a collaborative environment allowing a municipal assembly to work via Internet typical public sector software[16].

3.2 Reasons and base for public software and resource pooling

3.2.1 Pressures for efficiency

Citizens and politicians consistently call for increased accessibility and efficiency in the public services. Demands include a "one-stop shop" for all dealings with the public authorities and the "24x7" availability of paperless administrative services. are simple, user-friendly and adapted to all. Citizens would like to see their administrative tasks (as for example filling their tax declaration) processed almost automatically (based on the various information sources disseminated inside the administration), and just checking before signing; Citizens would like to receive automatically the various grants where they have rights, and not filling a great number of on-line web forms, etc. At the same time, citizens want to be the requesters and beneficiaries of such data collections, but will

16 Not all public sector developed applications are clearly at one side of the above borderline: is a web site content management system a "public sector" application if it can be used for all purpose, including eGovernment? We will not consider such CMS as a PSS

strongly protest if all these administration-stored data were merged or mined without consideration of their privacy.

The request for more efficiency ignore country borders: a British worker taking retirement in the south of France will request for pension data collection across several countries if he worked in UK, in Belgium and in Germany. All these interoperability requirements, combined with the complexity of the use of languages in government to citizen communication, open wide area for improvements and developments. Needs are growing faster than resources, and without using new methods, the software gap could grow exponentially.

At the same time, decision-makers are now more than ever aware of e-Government innovation by their neighbours and assign a high value to staying at honourable level regarding best practices. All these demands come at a time when the public sector is increasingly faced with funding constraints.

3.2.2 The policy of building the knowledge society

Meeting in Manchester on 24 November 2005 at the occasion of the Ministerial eGovernment Conference "Transforming Public Services" of the United Kingdom Presidency of the European Council and of the European Commission, the Ministers of European Union Member States, Accession States and Candidate Countries and Ministers of the European Free Trade Area (EFTA) Countries, responsible for eGovernment policy where unanimous to declare that transformed public services can contribute significantly to the achievement of the Lisbon goal of making Europe the most dynamic and competitive, knowledge-based economy by 2010.

In their approach, the innovative use of ICT by public sector can prove to be a powerful catalyst and a key enabler in transforming society as already demonstrated in many examples across Europe. The innovative use of ICT can also make a significant contribution to achieving Europe's sustainable development goals. All over the Manchester declaration, there is a strong believe that effective use of ICT can transform the life of Europe's citizens, businesses and the European economy in general, as it can also transform the public administrations themselves.

An instrument of this transformation is a clear, target-based European Action Plan for eGovernment that is required as to highlight successful policies and strategies and facilitate the sharing of best practices between all associated countries.

3.2.3 The expected impact on society

The impact expected is as follow:

- lead to significant improvements in public sector effectiveness and efficiency, both in customer-facing services and internal processes, including policy and implementation research;
- reduce administrative burden on businesses (especially SMEs) and citizens
- improve quality of life and can contribute towards trust in government and democracy
- Lower costs or spend with more results and return on investment
- Gain by cooperation across Europe and across policy areas in terms of research, experimentation, the sharing of good practice and delivery

3.2.4 Is common development (mutualisation) an option?

The idea of "sharing" is put forward at several place of the Ministers' declaration. Indeed pinpointing successes or best practice is only a first step in the improvement approach: in order to speed-up the implementation of these practices, a sharing, a reuse of successfully developed solution, or even putting resources in common to develop new ones is necessary.

Taking public e-Procurement as a leading case, the EU ministers committed to the capability of carrying out 100% of procurement electronically, with an effective 50% target. To achieve this goal, public administrations will have to do more than simply benchmarking best practices and reusing what could be: concerted efforts will be required in both the sharing of lessons learned and the development of eProcurement solutions.

This means sharing technologies and also, where appropriate, "developing common solutions" and working towards interface harmonisation of existing solutions. Once a solution will be available, this means also a development of the means of dissemination across all levels of the public sector (for example, a repository where knowledge and software could by found, licensed and downloaded).

Mutualisation may present obvious advantages concerning the sharing of costs (due to the scaling affect) and concerning related services: the documentation, the training, the maintenance and future software improvements. On the other hand, mutualisation may present additional cost.

These additional costs are related to organisation, which could be difficult to set-up and maintain. The decision to mutualise is preferably taken "ex ante" prior to initiate the work, in order to define the needs in common. The alternative of "ex-post" mutualisation "by adhesion" implies that a proposed solution will be adopted by other afterwards, after possibly paying some participation to the initial costs, which is often difficult to negotiate.

Mutualisation also presents additional risks of mixing different needs by trying to provide some satisfaction to everybody and producing finally a solution that is heavy, with poor performance and finally not really adapted to the needs of anybody. Objectives must therefore be clearly defined, as the process to modify these objectives and to avoid forking as far as possible.

3.2.5 Where could Open Source help?

The Open source licensing model helps, especially in the case of mutualisation, because it solve most of intellectual property questions between partners by ensuring them "full and equal rights" concerning the produced software. Even if private partners participate to the project, the open source model generates a spirit of mutual trust. This is because all open source software licensing agreements have commonalities: the author (the "licensor") gives at least four fundamental freedoms to the user (the "licensee"):

- The freedom to study how the programme works, and the freedom to adapt the code according specific needs. Access to the source code is a precondition for this;
- The freedom to improve the programme (enlarge, add functions);
- The freedom to run the programme, for any purpose and on any number of machines;
- The freedom to redistribute copies to other users.

For public sector in general, and even outside the approach of pooling or mutualising, the above licensing model is especially interesting because the software produced by or for administrations

are usually not “industry packages” that can be used “as is” by other users. In particular, Europe is a territory of diversity (languages, regulations, cultures etc.), and a software developed in France for example, will not be usable as is in UK or in Sweden: the French administration's remit is to respond to the needs of its own citizens and not to make business with a generic product that can be sold “out-of-shelf” across the world. As a consequence, the reuse of such software is depending on the revision and the adaptation of its source code.

From these two prerequisites, the supposed absence of commercial goals involving licence fees and the necessity to deliver the software with its code in order to adapt it to local realities prior to implement and redistribute it, the idea to adopt the “Open source Model” comes naturally.

3.2.6 Where to find best practices, examples or guidelines?

Collaborative development of open source software will be the main focus area where the future IDABC Open source observatory repository (OSOR) could provide organisational and legal material. The OSOR possible directions will be clarified after a feasibility study.

In the meantime, specific projects are dedicated to mutualisation methods. The cases studies developed in the previous chapters illustrate options and methods. In the specific area of middleware, ObjectWeb.org is an open-source software community created by a group of public and private partners with the objective to develop open source distributed middleware, in the form of flexible and adaptable components. By federating not only developers, but also projects ObjectWeb's ambition is to implement a complex mutualisation environment, pretty much like plants in an ecosystem.

National collaborative environments, as ADULLACT.org in France combines information, a common server to mutualise projects, a software inventory « ready for downloading » (e.g. the management of collectivities), interoperability tools and demonstrations.

3.3 The selection of a licence

3.3.1 Licence types

As said previously, a software is a work submitted to the copyright legislation, and therefore its usage by any other person than the original author must be authorised by a licence. Creating a licence is not easy (it is the business of IPR specialists) and therefore the use of an existing licences is recommended.
The issue is the number of available licences: more than 100 today (available means that you are authorised to adopt these text for the licensing of your software). A significant part of these licences are published on the Web site of the Open Source Initiative.[17]
Why such a great number, if the four fundamental freedom above are present in all licences? The reason is mainly related to the modalities of re-distribution and to the control or feedback that the original author intent to keep on its work. In these fields, the diversity is nearly without limit.
Fortunately, only a few licences are really significant in term of covered project or potential: the selection should be done inside a group of 5 to 6. The main question to clarify is the distinction between two main families of licences: **copyleft** and **permissive**.
An open source licence is „copyleft“ (also called “Share alike”) when it imposes on each subsequent contributor or “re-distributor” the obligation to distribute the result as a whole as open

17 http://opensource.org/

source, and in specific cases, to reuse exactly the same licence. The most typical example is the **GPL** (General Public Licence) supported by the Free Software Foundation (FSF), that is currently adopted by a wide majority of the open source projects, even if some of the most famous projects use other licences. The initial choice of a copyleft licence other than the GPL, even if it might seem interesting at first sight, is not generally recommended if the project owner wants to benefit from the contributions of the wide GPL community.

At the contrary, permissive licences, also called "free for all" imposes few constraints in the case or re-distribution. In particular, there is no "copyleft" obligation, meaning that a proprietary software developer can include the distributed OSS components in his products. A typical permissive licence is the BSD (this type include also the X-11 and other X-type license[18] as Xfree86, the BSD and modified BSD, the Apache software license[19], the Cryptix General License[20], The W3C Software Notice and License[21], The Python Copyright License[22], the Zope Public License[23], the LDAP public license[24] and the Phorum License[25] just to mention a few of them.

There is an intermediary type, also called 'Keep open", where modifications to the software delivered under such licences have to be made available as open source as well, while larger works incorporating the software are not subject to the same conditions (meaning that they can also be distributed as permissive of even as proprietary software). The **LGPL** (for GNU Lesser General Public License) used for Linux system libraries is one of these. The Mozilla Public License (**MPL**) provide the same compromise between Open source development and business. The LGPL is identical to the GNU GPL in many ways, but distinguishes two different kinds of situations when one uses a library. A "work based on the Library" means either the Library itself or any derivative work under copyright law, while a "work that uses the Library" means a program that contains no derivative of any portion of the Library but is designed to work with the Library by being compiled or linked with it. This includes also proprietary software that could run inside the open source environment. Therefore the use of such licences is quite specific.

3.3.2 Guide for choosing a licence

Due to the "Copyleft" character, the choice of a licence is often determined by the presence in your software of existing components. For a public sector responsible wishing to distribute "brand new original" software produced internally, the license type choice is easier if he responds (by yes or no) to the six "questions on usage" below.[26]

18 http://www.x.org/xdownload.htm
19 http://www.apache.org/license-1-1
20 http://www.cryptix.org/docs/license.html
21 http://www.w3.org/consortium/legal/copyright-software-19980720
22 http://www.python.org/doc/copyright.html
23 http://www.zope.org/resources/zpl
24 http://www.openldap.org/software/release/license.html
25 http://www.phorum.org/license.txt

26 This grid was published in the "Pooling Open Source Software" study and was adapted from the quick reference for choosing a free software license, -08 http://www.zooko.com/license_quick_ref.html; for an interactive version, see: http://yoyo.org/~pgl/lqr/

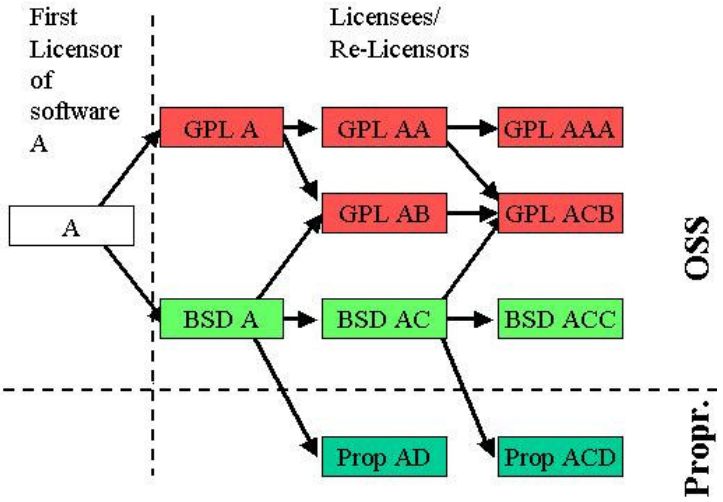
License	Q1 Do you allow other programmers to use and modify your code to redistribute it (as modified version or as derived work including all or part of your code) ?	Q2 Do you allow "Vendors enterprises" to combine your source code with their (proprietary) products and to redistribute these products (without OSS license)?	Q3 Do you allow other programmers to combine your source code with other "GPL licensed" source code, and to redistribute the whole under the terms of the GPL license ?	Q4 Are other programmers modifying and redistributing your source code obliged to publish / share the source of their modified or redistributed version ?	Q5 If the recipient combines your code with his own contribution and then ships the resulting combined app, that he must contribute a license to any patents that he holds that would restrict usage of the resulting app must include patent license with contribution	Q6 is your code based or including one or more pieces of code that was obtained under GPL License?
BSD	**Y**	**Y**	**Y**	**N**	**N**	**N**
LGPL	**Y**	**Y**	**Y**	**Y**	**Y**	**N**
GPL	**(1)**	**N**	**Y**	**Y**	**Y**	**Y**
Mozilla Public License	**Y**	**Y**	**(2)**	**Y**	**N**	**N**

(1) Some members of the industry software community refuse to accept GPL'ed source code into their projects, although other members of the community strongly prefer GPL'ed source code over other licenses. Contrast with code under BSD et al, LGPL, or Mozilla PL 1.1, which nobody refuses to accept.
"2" MPL 1.1 can be specifically amended to allow combining with GPL, according to the FSF's license list.

In the case of re-distribution, the difference is obvious: If the first licensor of **software A** delivers it under GPL, all contributor will have the same rights, but with no alternative concerning the sub-licensing: GPL only. If the initial developer (full copyright owner) chose a BSD licence, he put the code at the disposal of the community with almost total freedom: the same code may

- become proprietary in its version AD or ACD;
- be (modified and) re-distributed under the same BSD (or any other license) in version AC;

- be (modified and) re-distributed under GPL (and stay GPL) in version AB.



3.3.3 Which licence is most appropriate for the public sector?

The choice of a licence is closely related to the possibility of "appropriation" or "privatisation" of the software by a third party. Some stakeholders claim that PSS should be distributed under a "permissive BSD" licence type allowing everyone to reuse public material and redistribute it as proprietary one if opportune for any private business reason. On the other hand, other are in favour of a "copyleft" licence (GPL or LGPL) providing more guarantees that the code will stay open in all future versions, if any.

To highlight the question, lets compare with the law: there is no doubt that a State has the copyright on the law, on the case law, on its guidelines and government processes translating the law into operational transactions involving officials, citizens or enterprises. We acknowledge that it is the interest of government that the law, the case law, derivated regulations and guidelines should be distributed to all, without hiding any provision, without discrimination, exclusion or important fee to pay. This is the reason why such matter are increasingly published on the Internet for free. This should be the same with Public Sector Software, at a time electronic processes becomes the preferred way for the authority to interoperate with stakeholder: what we call e-Government. Similarily with the enlargement of the scope of e-Government and the progresses of the ICT based information society, it is important for transparency and democracy that PSS becomes publicly available and screenable, in order to ensure that the law is correctly translated into reality.

For the same reason that no one should appropriate exclusively the law, no one should appropriate exclusively public sector software. Does this exclude any distribution by private sector? Certainly not: they are numerous legal publishers, which publish the law in expensive books (or any other media including electronic ones) generally after adding their contribution: they coordinate the successive provisions in a single text, they add footnotes and their own explanatory comments, on which they have copyright. This distribution is not an appropriation, because any other publisher (and the government itself) could also re-distribute the same collection of law, case law or guidelines, however the added value (e.g. comments made by a private lawyer, by a university professor) are exclusive and "proprietary" because they are "add-on" that could be at any time separated from the original material.

Now the key question: could improvements to PSS becomes proprietary? The answer to this question will determine the type of licence that will be recommended for public software: permissive (like the BSD) or copyleft (like the GPL). As we already reported, the comments and added value provided by any academic author to the law, case law or government practices are proprietary: no one can re-distribute such works freely. However the original material (the law, the case law) is never modified. The proprietary added value (the comments, the articles) can be removed from the original without removing anything substantial to the "solution" that the law or case law bring to general or specific situations. The situation of software code may be very different: by improving it or modifying it, the derived work may become a new "one-piece" solution where no user is able to differentiate what is new and what was original, and where a return to the original version is not easy.

Sometimes because of its own statutes, forbidding exclusive appropriation of public assets by third parties and for principles reasons, the public sector will be in general attached to the perennial character of the open source specificity in the case where software produced by public funds is produced. For example, when the European Commission planned to distribute its own software (the IDABC software: CIRCA, IPM, eLink etc.) the received specification mentioned a number of operational considerations including , namely … "*protection against appropriation of application by third parties*".

As a consequence, the choice of a non-copyleft, permissive license allowing BOTH the proprietary software industry AND open source communities of developers to improve and fork their respective modified versions of public sector software under various sub-licenses (proprietary, GPL or others) was not recommended in the CIRCA case.

Therefore, a "copyleft" licence is generally recommended for PSS as a guarantee that the solution will never be "appropriated". This left open the possibility of separated improvement or "plug-in" provided by industry under the form of separated objects that may be distributed under other licences. To benefit from the support of the widest developers community, one choice is obvious: the **GPL** (currently version 2) **LGPL** or alternatively a copyleft licence that includes a clear compatibility clause with the GPL (as the EUPL discussed below).

3.3.3.1 Dual licensing

Another option is to use more than one license. The copyright holder has always the freedom to license his work under several different licences, depending on the target and the use (e.g. proprietary or not). This is called "Dual licensing". Indeed, software released by public bodies could be used to different sectors, depending on its application domain. It is possible for public bodies to "have their cake and eat it too" through dual licensing. This involves releasing the software to the general public, including all other users and developers, under a reciprocal licence for the benefits of such a licence. In addition, the software is available for licensing to developers who want to create proprietary derivatives, under separate, non-open-source (or "private"), licensing terms. These could be commercial terms involving royalty payments, or allowing the public body to buy derived works at a discount. Or these could simply require that the developer notify the public body before making proprietary derived works.

Dual licensing is a powerful method of carefully aligning the publishing terms to policy goals (see Table 11) and is feasible for all works (or components) that are fully owned by public bodies.

Indeed, dual licensing is a strategy adopted by firms who release software they fully own, the most successful example being MySQL (which releases software under the GPL to the public, and under proprietary licenses to other firms that want to make proprietary changes to its popular database engine).

Policy aim for publicly funded software license	Copyleft	Permissive	Private
1. Initial software product available to the general public for use and modification	Strong	Yes	n.a.
2. Further developments available to the general public for use and modification	Strong	No	n.a.
3. Original developer can exploit product under proprietary licence	Yes	Weak	Yes
4. Third party developers can exploit product under proprietary licence	No	Strong	Yes
4a. Third party developers can exploit product under proprietary licence without approval of original developer or public authority	No	Yes	No

Impact of publishing licence on policy aims[27]

3.3.3.2 Copyright dispersion issues

One argument that is sometimes presented against copyleft and dual licensing for publicly funded software is that the copyright of the software product is rapidly dispersed as new *community developers* add their modifications to the software, and remain the copyright holders for such modifications. If this presents a problem, it can be solved by setting up a simple structure based on dual licensing that allows this, using funding conditions rather than licensing conditions to vest copyright within a single copyright holder that can license out the software to third parties under a private license, while licensing the software to the general public under a copyleft license.
When a contribution to the software, after its release, is made by an agency receiving public funding that falls under this scheme, that agency is in effect a *funded developer.* Although the funded developer is a copyleft licensee of the software, and the modification is made under the rights granted by the copyleft license, the funded developer differs from a community developer from the general public in receiving public funds. Funding conditions, therefore, can require what FS/OS licensing conditions do not – the transfer of copyright over contributions to the central agency. It is important to note that this copyright transfer is a result of funding conditions to the funded developer, and does not require a special license. A special license would probably not be FS/OS, let alone copyleft, and would strongly discourage the involvement of members of the general public in further development and use of the software product.

This structure results in the copyright over all contributions and modifications to the software from entities that receive public funds being vested in a single copyright holder, the *central agency.* This copyright holder could continue to license out the *publicly funded* software to third party developers for distribution under a proprietary licence, and distribute any fees or royalties received back to the public funding system (or indeed, to the funded developers that have contributed to improvements to the software). This central agency would not be able to license out versions of the software based

27 Source: Ghosh 2003. Reciprocal e.g. GPL, EUPL; Permissive e.g. BSD, Apache. Private = non-open source licence (e.g. proprietary licence requiring royalty payment or other conditions).

on contributions from *community developers* from the (unfunded) general public. However, since the software is distributed to the general public as well as funded developers under a unified copyleft license, everyone – the general public, the research community and all the funded developers, as well as the original developer and even third party developers – would be able to use, modify and otherwise benefit from, though not appropriate, such contributions.

3.3.4 Would a EUPL be better adapted to European realities?

Most of the available licences are written in the United States. On one hand they are convenient for use in Europe, because a wide majority of European developers use it (especially the GPL) and because the copyright aspect, which are essential, have been declared valid by some European courts in the case of the GPL.[28] But problems might arise for the following reasons:

- Only the English version has any official value (translations are provided for information only), while the European Union and Member States are entitled (or have sometimes the obligation) to obtain official value in other languages
- US licences present often a "wild" text structure and vocabulary which is unusual in Europe
- Specific European applications of copyright (the right to "communicate to the public", the scope of moral rights) are ignored
- Liability exclusion is too wide (especially regarding some Member States law as Germany that are not accepting general exclusion "to the extend permitted by applicable law")
- Same remarks for warranty exclusions – without justification
- The mandatory character of local regulations (e.g. national laws applying EU directives for consumers information and protection) is ignored
- No clear determination of applicable law[29]
- No clear determination of competent court
- Some licences explicitly refers to US applicable law or to US competent courts.

For the above reasons a licence which is perfectly convenient for developers will not be convenient for public services. This was the case at the European Commission, which developed its own draft licence, the EUPL. By developing its "own" licence, the Commission was also interested to stay "the author" of the dispositions related to its own software and to avoid dependences from any other body: the current discussions and disputes related to the future adoption of a version 3 of the GPL illustrate how difficult the revision of existing provisions may be.

Once it will be officially approved and used by the European Commission, the EUPL could be an alternative because it will have an official value in several EU languages (starting with the three working languages) and because more in line with European context and understanding. However, to be "accepted" by developers the EUPL has to be compatible with other copyleft licences as the GPL. Therefore, EUPL (from V0.2) contains both a copyleft and a compatibility clause.

The purpose of the compatibility clause is both protective and flexible: the public administration may determine which licences are compatible. These will be selected among other copyleft licences and sould not include permissive ones. The GPL will obviously be placed on top of this list. The

28 German Netfilter court case (District Court of Munich I, Judgment of 19/05/2004)

29 Or referring clearly to an US jurisdiction, as the MPL's applicable law and forum clause, referring to California.

well understood constraint is that, once declared compatible, a licence could not be removed from the list, otherwise no legal certainty would exist. Another advantage is found in maintaining the application of the EUPL as far as possible: a redistribution under the GPL licence will be authorised is the software has been improved or complemented by the inclusion in (or of) a licensed GPL component, and not in other cases.

3.4 Check list before publishing Public software

Prior to any distribution, it is necessary to ensure that the software is:

- Corresponding to a general need and commitment?
- Full and exclusive ownership of the public administration?
- Suitable for external disclosure and publication?
- Developed according to the open source model?
- Suitable for licensing?
- Suitable for obtaining long term support from a community?

3.4.1 A general need and commitment

At least some preliminary fundamental questions should be investigated, and each of these questions corresponds to a key factor of success. The first question is related to the evaluation of the need: is there really a common and actual need for such software in several administrations or similar user groups, in order to create a community of users and contributing developers?

A second question is the lucid evaluation of “your participation”: why should your software be the best candidate to respond to the need. Is your ambition really to provide the community with a generally usable and adaptable software toolkit (and benefit in exchange from other software contributions) ? Or is it just to solve your own problem: e.g., you cannot maintain the software anymore, it is too big or it costs too much, it is not well documented or initial developers disappeared and you hope that by giving it as an OSS it will immediately benefit from free maintenance and upgrades?

Concerning the commitment evaluation, what will your future contribution be ? Will you be able (or developers in your administrations) to actively participate in maintenance, to distribute the work among interested contributors, and to provide for and foster the exchange of knowledge related to the software and its development?

3.4.2 Ownership

If the software is entirely produced by the PA (e.g. written by PA employees during service hours) without reusing components found outside, we can assume that the PA has a full ownership on it. In most cases, the situation is not so transparent: if parts of the developments are done by contractors, it must be clarified if the development agreement transfer full property rights to the PA, with the right to redistribute it without any limitation.

Quite often, development contracts contains a declaration of “transferring ownership” or “exclusive ownership” to the PA, meaning that the contractor transfer all his rights to the PA and will not re-sell the same solution. Such contract should be reinforced by authorising public disclosure of the code (and granting the PA against actions in the case of copyright or patent infringement) and authorising public licensing of the code by the PA (and granting the exact model of licence that could be used then, depending on the possible licences of the components imbedded in the software).

3.4.3 Suitable disclosure and publication

The status of the software maturity must be clearly stated (preferably by external experts). A lack of maturity is not a definitive obstacle, provided the improvement requirements and steps are clearly identified.

A self evaluation of this maturity and of possible related risks is obviously very difficult to process internally. It is the job of specialists. The simple fact that a solution "has worked well during months in a specific environment" does not mean that it will operate elsewhere without serious problems. It means even less that it could be certified for sensitive application, for the management of privacy related data, in the context of defence or for the protection of critical infrastructure.

By chance they are specific projects where, aware of the issue, specialists are going to set-up environments dedicated to the quality of open source software. We might mention for example the CETIC in Belgium or the QualiPSo Integrated Project, where the aim is to implement technologies, procedures and policies to leverage the Open Source Software development current practices to sound and well recognised and established industrial operations.

3.4.4 Developed according to the open source model

It has been said that the real core of an open source program is the licence. However, the practice demonstrate that associating an open source licence to a massive piece of code does not make automatically a successful OSS project. Even if there is an acknowledged need, shared by many groups and communities, the take over may as well take years or never happen. It useful to remind here some success factors related to the OSS development model.

An usual open source prerequisite is common development: Programmers (starting from a single person to building a community of developers) have developed an interesting working solution to respond to their own needs or problems (because no solution was available for their environment, or because the existing solutions were too expensive).

The question we need to answer is "How far, or on what conditions does software developed by an administration correspond to the open source development model?" Are these software a good basis for open source development?

The main conclusion is that applying an OSS licence on existing software to benefit from free maintenance and upgrades is an illusion, if the software does not respond to precise development conditions:

- A reasonable number of persons "sharing the same problem";
- Initial but flexible repartition of "ownership / leadership" between diverse persons, from diverse organizations;
- A comprehensive and well structured documentation;
- A roadmap (navigate into the code as in a web site);
- A common trunk easily understandable + functional modules / no monolithic code;
- A code organised in many relatively small pieces (in order to facilitate individual take over);
- Clear identification (and communication) of what parts are "mature/stable" and what parts are "to improve" (according to the "release often" principle);
- Permanent discussion forums on requirements, objectives, priorities for further

development.

In consideration of all these conditions, the fact to “go open source” may represent for the public sector an initial investment (existing software) or a serious development cost augmentation (for making new software) that will be recovered long term.

3.4.5 Suitability for licensing

If the code is to be written entirely written from scratch, we assume that it is suitable for licensing to third parties and that the PA is entirely free about the choice of a licence. In other cases, since most solutions are based on integrated components, it is necessary to obtain full details of the licences related to these components.

In the case of mutualisation and in all cases where an association of several developers is established (for example, the cooperation with an open source community), it is opportune to clarify aspects that are not regulated in the licence, such as structure for decision-making, questions of representation and the sharing of costs.

In the case of contracting all or part of the “open source” development to a service provider, the convention should include dispositions about the assignment of all rights to the contracting public body, the determination of interoperability standards (or at least a commitment to open standards), and the licence compatibility of the components used in development.

If the code is to be written in collaboration with an external open source project, a number of other considerations apply. A close collaboration between a public administration and an existing open source community may be highly recommended, because the organisation of the work will adopt the open source organisational model from the first moment (collaborative environment, granularity, mutual screening etc.). Such collaboration will provide early learning on the potential success of the solution and its perennial character: Are they indeed so many people sharing the same issue? Could we really believe that the community will grow fast enough to provide support? Will it make the solution self-supporting and financially independent long term, as the purpose of the public administration may be to provide initial start-up investment, but not to be committed to invest forever.

In the best hypothesis, the PA is will build on an existing growing “success” by looking for using and adapting an existing OSS project, that has already a dynamic community of developers (and if delivered, of users), responding to a specific purpose (e.g. a content management, a collaborative work environment, a work flow system that could be used by various PA but also for other business).

After identifying the community and its leaders and copyright owners, and checking if the distribution license is compatible with the PA needs (negotiating appropriate license or dual licensing is often possible), an agreement could deploy contractual or extra-contractual incentives, such as a service level agreement or bounty awards for specific programming tasks.

If the PA has already developed a solution with a specific public sector purpose (e.g. land record management, public health) and wants the solution to be take over by an Open source community for ensuring perennial character, financial independence, improvements and support. Prior to the decision of going open source, the PA should take contact with national / European administrations that could share the same needs (and with their associated free developers organisations, clubs or communities, if any) to evaluate the interest of the distribution (see if no alternative exists and if it could not be more profitable to join another project).

The conclusion of this approach has to be honest and avoid the temptation of the NIH (not invented here) reaction, which is not easy when a responsible is naturally proud about the realised solution and would like to promote it as a standard.

Each component of the solution will be carefully screened regarding its licensing conditions: it is not unusual that some components that were functionally good will have to be changed or rewritten just because their licence was proprietary or non compatible and because no agreement was found with the copyright owner to modify licensing (it is always recommended to request, as the copyright owner may have the option to provide dual licensing).

Initiating a project with a fair part of the solution is an advantage. At the contrary, because the code was not developed from scratch by a community, there is no guarantee that a new community will be interested and endorse the project. A strong incentive policy may be necessary.

A solution to initiate and obtain a certain guarantee concerning take over and support is to contract long term (2 years or more) a service level agreement with a service provider at the same condition as for the first hypothesis. A part of the service provider mission will be to animate a contributor's community and set up the appropriate tools for it (therefore, chose a partner having experience in such co-operation, and contractual conditions imposing him to be open and to set-up the corresponding technical environment). The cost will be higher than for the second hypothesis, but obtaining a result will be more simple and secure, providing more guarantee to generate long term a successful open source product supported by a community of volunteers.

3.4.6 Suitable for obtaining long term support from a community

One of the main attributes of open source software is not the cost, nor the openness of code in itself, but the ability to freely share knowledge among a large community of collaborators. Public Administrations may want to benefit from this. Guidelines for Public Administrations on partnering with free software developers include a step-wise guide, a background briefing on motivation, organisation and funding models for developer communities, and legal frameworks for collaboration. The main 11-step guidelines for collaboration are summarised here.

Identify the "seed" – what will attract developers?

Show what you've done before asking the community to help. The best way is to release the core of an application, developed in-house or by subcontractors. This provides the seed around which a community of developers can form. A group of PAs can collaborate to do this – see the article by Oliver Schneider in this issue of Synergy for more on inter-PA collaboration.

Disseminate to reach out to the developer community

If nobody knows of your needs, nobody will respond to them. If you develop "seed" software but potentially interested users and developers (including other PAs) do not learn quickly about it, a community cannot form. Widespread but targeted dissemination is essential

Attract a community to solve a problem

Attracting an existing community can work if your problem is shared by others. You still need to follow the previous steps. This is how GNU/LinEx in Extremadura, Spain succeeded in attracting the broader Debian developer community.

(or) Attract a community to support a pre-existing software application

An alternative to (3) is to release a fully developed application of wider interest – planting a tree rather than a seed, with the community invited to improve rather than create the solution.

Provide or identify collaborative development infrastructure

PAs can provide themselves with channels to reach developer communities by creating development infrastructure, portals and repositories that provide useful services to developers.

Cooperate and proactively provide feedback

Users form the basis for development in the free software model; therefore it is important to ensure that users in PAs interact extensively with developers. Free software developers need feedback, and they usually act upon it quickly.

Identify community leaders

While free software communities are self-organising, leaders are essentially self-selected in that those who show the most initiative become de facto leaders. It is useful to try to recognise such individuals and interact extensively with them.

Identify selection mechanism to balance competition with cooperation between developers

While portrayed as cooperative and communal, free software development is in fact a form of Darwinian competition. PAs need to provide mechanisms to actively influence the selection of competing solutions.

Identify funding methods if required

Developers are not volunteers providing charity. PAs must have clear strategy for generating developer motivation for collaboration. One motivation is financial, and there are various funding models possible.

Monitor and evaluate results

Continuously monitor the extent of interaction between “client” PAs and the “supplier” communities.

Involve other public administrations

A community of supporting developers is attracted by an existing community – e.g. of collaborating PAs. Single PAs are much less likely to succeed than a group of PAs in generating a community of supporting developers.

3.5 Barriers and questions on distributing Public software

3.5.1 Public assets disposal

A first possible objection is: Is distribution compatible with budget regulations? Can the public sector give assets for free, especially when these assets are possibly transmitted to “non-national taxpayers” who have not contributed by their taxes to the public sector budget?

Software presents differences to other public assets due to its immaterial character: A software is a work submitted to the copyright legislation, and therefore its usage by any other person than the original author must be authorised by a license. If the copy and the use of a software is licensed by a public authority, there is no “deprivation” in a sense that an intangible, numeric software code can be duplicated or downloaded an infinite number of time without loosing any substance or quality.

Also, there may be a direct interest for public sector that the widest possible use should be made of its software: the more a software is used, the greater is its value and the higher is the expertise of both users and service providers, related to this software. The best opportunity for any software is to become a world-wide standard. Distributing its software rises therefore the value of it: more users, more experience and technical expertise, more possibilities of improvements, perennial character, choice of various service providers. For public sector, distributing software demonstrate a wise usage of public money.

As previously analysed, a typical Public Sector Software is essentially the translation of the law (in the large) into ICT operations, and therefore, due to the growing importance of e-Government, the widest possible transparency, distribution and screening is even to recommend and is in “the interest of the nation and of democracy”.

Depending on the choice of an appropriate licence (see hereafter), software distribution is therefore not misusing public budgets, however this may appears as “forbidden” by some national regulation: a preliminary assessment of national legal obstacles is recommended. If needed, a legal clarification could be adopted concerning the specific case of software.

3.5.2 Competition with the industry

Is administration acting as software distributor competing with the private sector? It will undoubtedly have an impact, and may lead private sector to adapt its proposal and business model accordingly.

Though the publication of software by public bodies may constitute competition for business activities, this competition is not unfair. In the eighties and early nineties, entrepreneurs set up commercial databases of legislation and case law, building their enterprises on texts produced by the government. When, towards the turn of the millennium, governments started publishing legal texts on the Internet, the basis for this business disappeared; but the publication of legal texts by the government was not seen as unfair competition with the database industry. Business is a constant adaptation to new realities and the new government policy was just one of these.

Concerning the distribution of its own software (where the public sector also owns the copyright), the impact would be less dramatic and brutal: Specialised public sector software represents only a very small fraction of produced software, and even a small fraction of software used/purchased by public sector. In addition, even if some public sector software projects present all the necessary characteristics “to be distributed”, the number and variety of obstacles (legal, financial, organisational, quality level) is so high that only a small minority of candidates will finally be effectively distributed, this being done very progressively.

At the same time, the publication of Free Software by public bodies will create numerous business opportunities. These range from contract work to prepare a software package for release, to localisation for other languages, and the integration of complex software solutions.

3.5.3 Confidentiality requirements

Confidentiality may be another obstacle on the road for public sector distribution. Publication advocates will put forward the principle of “Security by transparency”: when you publish software, it does not make it less secure, at the contrary most possible weaknesses are cross-checked by thousands eyes and are therefore more rapidly reported, analysed and repaired. At the contrary, hiding weaknesses (“security by obscurity”) by the fact they are not disclosed publicly is not a valid long term guarantee: soon or later someone will find the back door, and possibly this back door had been introduced and hidden deliberately by some developer.

On the other hand, the sudden disclosure of a large number of code lines by public administration may also suddenly disclose weaknesses that will be immediately used by hackers to penetrate the system prior it could be secured. Transparency may therefore be secure long term, but insecure short term.

We are here typically facing a case to case situation. We do not believe that public sector software is more secure by nature, or that the open source software is more secure: code may be good or bad in all cases and non disclosure may be the most appropriate short term decision in case of doubt (so long the software has not been screened and certified by a competent body), especially if it provides access to sensitive personal data or to information whose disclosure may endanger the essential interest of the State, of justice, of the citizens or enterprises.

3.5.4 Procurement requirements

The public procurement process is extremely regulated and their purpose is to place all competitors on an equal footing. In that sense, procurement rules aim for competition, not for collaboration: the may become a major barrier by making collaborative development difficult, especially when partners belongs to different jurisdictions.

When calling for external services and in the presence of two distinct business models, the open source (based on services on an open world) and the proprietary (based on higher initial licence fees followed by a more or less exclusive interdependency between the vendor and the client), the public administration should not prioritise one model against another.

For these reason, technical specifications releases together with a public sector call fro tender should never include wording as “ we want an open source solution”. In addition to discriminating a category of product or one of the business model, this would have no precise meaning, due to the great variety of licences. To be neutral, the public authorithy should screen the 10 conditions produced by the opensource.org to make a licence “open source” and select only the conditions that are objectively requested for the purpose of the task to deliver. By expressing these condition that should correspond to business realities, the market (all industries) will have the freedom to answer or not with an appropriate licensing model. Any mention of a proprietary environment (e.g. “the solution must operate in an Oracle context”) must be justified by business realities.

4 Conclusions from the study: Characteristics and impact of OSS releases by public bodies

4.1 General characteristics

The study identified following factors as main motivators for public bodies to develop and distribute own software as OSS:

- contributing to the OSS community (sharing the philosophy of OSS)
- enhancing service quality
- vendor-lock ins (i.e. the wish to become independent from software vendors)
- need for technical solutions that are not provided by proprietary software
- cost savings (savings of license fees, keeping old hardware longer, synergy effects through reduced training efforts, etc.)

In our survey, the motivation to "give something back" to the OSS community came out as the most prevalent reasons, indicating that persons who initiate or participate in such projects within public administrations may have ties to that community.

Except for Beaumont Hospital, which had to react to severe cost-cutting in its IT budget, each of the projects we examined had as its starting point a need for a functionality that was not provided by existing software or IT infrastructure. This observation complies perfectly to the theoretical considerations of increased usability through increased numbers of "vendors" an software products, as described in the theoretical model for OSS usage.

Given the fact that many PAs own software because they purchased it together with full ownership rights or because they contracted the development to a software house, that the number of cases in our survey that report experience with releasing OSS is appears quite a bit low, we can nevertheless state the following as tentative conclusions from these results:

1. the share of local governments that *are aware* that they own software that can be released under an Open Source Software is probably quite low (approximately 10%)
2. the share of local governments that own such software and has full rights on it is probably higher than the share of those local governments owning such software without having full rights on it, regardless of whether they developed the software themselves or contracted the development out; however, awareness of the existence of such software may be low, or such software may often be scripts or components that, while potentially useful to others, were not considered by our respondents as "software that can be released".
3. there is s apparently a potential for future software releases under an Open Source license among the vast majority of governments that currently do not own such software
4. the type of software predominantly developed by or for governments is neither general-purpose nor public sector specific software, but specialised software such as content management or work-flow systems, useful to the public sector but also to other organisations.

4.2 Organisational challenges

Developing OSS is an organisational challenge for any IT department in the public sector because it requires a lot of effort to become familiar with the principles of OSS. Moreover, it implies legal

challenges (license and copyright issues as well as liability issues) and some modifications of government procedures, as tasks since tasks like quality assurance, communication to the wider OSS community, building of technical steering committees and user groups etc. do not belong to the typical set of tasks in public administrations. Nevertheless, all case studies have found ways to meet these requirements and all use to a certain extent “community-like” developer bases, which means that they organise forums that interact via a website, but in contrast to huge OSS projects like Debian there is no community of real volunteers. The communities created and deployed by the public bodies we examined consist mostly of people who are involved in the project through work.

We did not find significant opposition or typical groups that were resistant to the OSS activities in the public bodies we examined. By nature, users are reluctant to changes in the beginning, but get adapted to these changes over time. Only in one case study, Beaumont Hospital, user resistance together with concerns of the management regarding the IT department's speed towards OSS led to a setback.

There is experience of proprietary software companies lobbying against OSS, but this is hardly observed directly at the public bodies we selected for case studies. Apparently, lobbying against OSS takes place on the higher levels of the administrative hierarchies in the public sector. Apart from that, some of these observations appear to be – in part – a reflection of a general distrust towards superior hierarchical levels or departments that are responsible for general decisions on budgets or purchases.

4.3 Personnel / skills requirements

Skills requirements are the bottleneck for PAs to become able to develop and distribute OSS – and thus the crucial factor for such projects to have a significant impact on society (see next section). Except for LinEx, all OSS projects of public bodies we examined in case studies were initiated by the IT departments of these organisations. This observation, which is in line with previous findings from other surveys, suggests that IT – or more specific: software-expertise is a very important precondition for public bodies to move towards OSS. This expertise must not be OSS expertise. As the examples of LinEx and Beaumont Hospital indicate, OSS projects can be quite successful even if no knowledge of OSS is available within the organisation when the project starts.

Though skills are the most critical success factor for OSS projects in PAs, it appears that OSS also provides the solution for this problem: All representatives of the case studies reported an improvement of the skills base at their departments through their involvement in OSS project, especially with regard to software engineering skills. OSS apparently provides a learning environment that enables public bodies' IT departments to learn within a reasonable period of time how OSS works, how it can be developed and help to find solutions for the special functionalities that are needed, and how it can be distributed and maintained. Thus, starting to deal directly with OSS would be the best way to overcome skill barriers that may hinder OSS projects in many PAs.

4.4 Economic aspects

Despite the fact that saving costs is a strong driving force for public bodies to migrate towards FLOSS, hardly any public body we examined was aware of the exact costs and benefits of its OSS activities. Reasons for this are that some of these costs or benefits are hard to calculate and that

often the OSS project appeared as “necessary anyway”, so that its results were needed more or less regardless of the costs of some of its parts. Overall the balance that is made by the case studies is however very positive in all cases.

There seem to be spill-over effects to local training and consultation agencies and to the OSS community, but they appear limited. The same applies to direct employment effects of the projects we examined. We think that by now public sector OSS activities only begin to have an impact on local economies, but that significant impact on businesses and employment could be reached if there was a broad take-up of OSS activities in European public sectors.

4.5 Technical challenges

Developing OSS is a challenge for any IT department in the public sector because it requires to become familiar with its principles and standards, which differ completely from just buying and using (proprietary) software, and implies some modifications of government procedures (such as quality assurance, communication to OSS community, building of technical steering committees and user groups etc.). Especially public bodies that have no programming capacities in-house have to find solutions to get the software they need developed. Another factor that determines the degree of technical problems is the complexity[30] of the OSS project and the overall time line in which it shall be realised. APLAWS and MMBASE provide surely the two most complex projects of our case studies.

A major point of concern when public bodies develop their own OSS is to assure good technical documentation. Again, complexity and time frame play a role here, but also the degree to which a project can be planned in advance and to which work can be distributed. As APLAWS shows, even a very complex OSS project can handle documentation requirements if this task is observed right from the beginning and if there are partners with the experience to take over this responsibility.

As a matter of fact, none of our case studies showed any indication that technical problems could have been so strong that their project could have failed. Just in contrast, we often found state-of-the-art development models as well as state-of-the-art OSS products in our case studies. Technical problems may decrease if in future more in-house software development capacities are built up in European public sectors.

4.6 Legal aspects

Most public bodies seem to become more familiar with copyright and licensing issues once they have decided to develop OSS. But the legal knowledge they obtain usually is not sufficient to give advice to others; they also usually do not ask for legal advice, which may imply that there is a lot of uncertainty. Public bodies tend to opt for GPL because it's the best known and “easiest to understand”, as one interviewee explained. One case (MMBASE) got a lot of problems because the management decided for a problematic license. Overall, license and copyright issues play a role, but our impression is that the interest to get deep into this topic is quite limited. Measures in order to provide public bodies with legal advice might be useful in order to increase public sector OSS

30 “Complexity” does not only refer to size, functionalities, etc., but also to the product that is developed. Adapting an existing product to the needs of a public body is *usually* less complicated than creating a completely new product. This is however not always the case, as the examples of the CMS developed in Camden or at VPRO show. Though there have been CMS, these turned out to be useless for public administration of a municipality or for a big broadcaster.

activities in Europe. However, people have mixed feelings about a new European OSS license, such as the EUPL. Developers and those closely involved with the developer community tend to be negative because they think that the existing variety of licenses is already enough or even too much (such people typically prefer the GPL). On the other hand, PA officials recognise that their legal services would appreciate such a licence, and that this may lead to the release of more software owned by PAs as open source.

4.7 Impact on government services

Given the results on incidence and acknowledging the fact that OSS projects at public bodies are rarely driven by considerations regarding the local economy, one must conclude that the impact of public bodies making their own software available as open source on government services and the information society appears to be quite limited. Obviously, OSS activities of public bodies have (and often are intended to have) an impact on the internal organisation and procedures as well as on technology use rather than on the information society as a whole. A closer look at the different contexts and conditions under which our case studies have realised their OSS projects, however, provides a perspective on the *potential* impact of in such projects and for different scenarios in which this potential might unfold in future.

Interoperability and transparency are usually considered to be bound to a broad set of political and technological requirements and objectives that cannot be reached by “simply” developing and publishing OSS. As main measures for interoperability and transparency we consider the degree (intensity) of cooperation between different public administrations (PAs) and – where appropriate - the existence of public-private partnerships in provision of an eGovernment service. The latter is relevant because public-private partnerships require collaboration of public authorities with externals and this aspect highlights the high requirements that come already into existence when only collaboration with businesses shall be realised. If interoperability and transparency in interaction with the individual citizen becomes an issue, these requirements may even increase.

IDABC's “European Interoperability Framework for Pan-European eGovernment Services” provides a number of recommendations for interoperable eGovernment services that provide insight in the complexity of this issue. In IDABC's concept, interoperability must be provided in the organisational and semantic as well as in the technical dimension. From this approach IDABC derives a number of requirements that should be met by eGovernment services:

- the service requirements should be jointly determined by the participating administrations via a demand-driven approach
- all involved partners should agree on Business Interoperability Interfaces (BII) that enable the business processes to interoperate across administrative units (which may require studying the definition of common BII standards)
- there should be service level agreements in order to formalise mutual expectations from the service providers
- the service providers should agree on a common security policy
- the data elements to be exchanged should be made interoperable
- the service providers should find a single language, based on XML, in order to make (legal) vocabularies interoperable
- technical interoperability should be provided at front-office level as well as at back-office level
- the technical operability of eGovernment services should be based on common guidelines

- these common guidelines should be based on recognised open standards

4.8 Impact on the information society

Due to the fact that OSS development activities of public bodies are bound to a need for special functionalities, the scope of impact of such initiatives is "by nature" limited by the scope of the usability of the developed software or the scope of the infrastructural change that is intended to achieve. Since most of the desired functionalities come into existence in the context of sub-tasks that are performed within PAs, it is not to be expected that OSS activities of PAs *per se* reach a societal scope. Rather than that the opposite appears to be likely. The fact that the local economy (i.e. the "first step" outside the boundaries of a PA, still far from the realm of "Information Society") is clearly not what PAs have in mind when they think about software development projects is a very strong indication for the limited societal impact that is borne in public bodies' OSS initiatives. Nevertheless, it is also obvious that the public sector just started to become an active OSS developer, and there is an opportunity for such initiatives to spread and to be performed by ever more PAs if there is a need for special software functionalities. We see a great *potential* for a fundamental impact of public bodies' OSS activities if the context conditions under which these realise are considered.

In the case studies we could differentiate between three different types of OSS development and release:[31]

- "externalised customisation": the public body has a need for software but no capacity to develop, distribute, and maintain it, therefore the development (of the first release) is completely contracted out. But requirements are clearly specified by the PA and the PA becomes the "owner" of the software and plays a vital role in maintaining the project after the first release is published. This type seems to be mainly driven by general political strategies, the software (OSS) itself appears as a means that helps to achieve more general political objectives.
- "experience-based innovation": the public body has a need for software and the capacity to develop, distribute, and maintain it It holds a position as coordinator over the whole project period and tries to ensure that erroneous developments or forking do not happen. This type seems to be mainly driven by a demand for (a) particular functionality (functionalities), the software (OSS) is not means for other objectives but objective itself.
- "cold adoption": the public body has a need for software but neither capacities (or the willingness) to develop this software in-house nor the financial means to meet the demand with proprietary software. Therefore the inexperienced IT staff has to attain programming and software engineering skills within a short period of time. This type seems to be driven by scarcity of resources (either financial or organisational), OSS is chosen in order to secure the "survival" of the organisation (or department) and / or the services that are provided within and by the PA.

Each of these three types of OSS development in PAs bears a distinct potential impact on the Information Society:

- "externalised customisation": Only the provincial government of Extremadura set objectives

31 We don't claim that this typology is exhaustive. It's main purpose is to serve as a starting point for the development of scenarios for the impact that public bodies may have on the Information Society when developing and releasing own software under an OS license .

with its OSS strategy that go far beyond the scope of the demand for special functionalities, as the LinEx project was integrated into a general "Information Society Strategy". As a well-known fact, the project resulted in a major improvement of the IT infrastructure in the public sector, especially the educational sector, and in improved computer literacy of the population. It is this "embeddedness" of public OSS development activities in a broader political strategy what characterizes this approach and what enables it to achieve significant impact on the Information Society, though on a regional scope only. The political element overcomes the limitation of the special need that the software is required to meet and transfers the impact of OSS from the level of its direct application to the social context in which it is applied. It is noteworthy that the limitation on the region in the case of LinEx does not derive from the OSS development project, it is set by the political strategy. As a consequence, if such development strategies are embedded in political strategies that target the Information Society on national, multinational or even global scope, it can be expected that OSS turns out the similar impact on larger scope.

- "experience-based innovation": The potential impact of OSS in this scenario is much more bound to the underlying special functionality that initiates PAs' OSS projects. A significant impact of public OSS projects that correspond to this type is only likely if the install-base of the developed software exceeds a critical level and becomes more or less a standard application. Probably, in order to have a "societal" effect, a certain degree of community involvement must also be reached. APLAWS as well as MMBASE appear to have the potential to become such "standard applications", but they also demonstrate how difficult it is to involve and affect actors outside the close user community in the PAs and other organisations that develop and / or apply it. NFI software like TULP2G, which also belongs to this type of public software development projects, is even more limited in this respect than APLAWS or MMBASE because this software serves mainly the demand of small niches in the software market.
- The scenario based on the "cold adoption"-type is flawed by the fact that the starting conditions of OSS development projects of this type are quite poor. This type is characterised by the fact that the staff has to catch up and attain new skills before any code can be written. After this process is completed, migration of existing systems rather than developing new software becomes an issue. However, once the migration process is completed the development process would be converted into the "experience-based innovation"-type, with all its limitations and potentials.

4.9 Considerations for a successful Open Source strategy

For public bodies looking to implement an Open Source strategy, it is important to understand the opportunities, but also the risks and possible problems related to an OSS approach, and to build internal support for the strategy. It is necessary to identify possible building blocks for the solution, as well as ways to integrate them. It is also important to verify that the intended addressees of the program are actually interested in it; if the organisation is ready to enter into the long-term commitment of maintaining a Free Software project; and if it is able to handle the technical complexities. Then, medium- and long-term strategies should be defined, using outside help where necessary. For public bodies, the priority is to deliver high-quality, reliable and continuous services to citizens and firms. Any implementation of an open source strategy should be carefully designed to support this primary mission. Good change management is an essential element.

Establishing competence centres for public bodies wishing to release their software under a free licence might be helpful. One task of such a centre would be to provide advice with regard to legal questions, in particular those concerning the choice of licence. It could also aid public bodies to manage redistribution and collaborative development of their software. Going further, it might provide Intern tools and collaboration services for public administrations involved in developing free software. Existing organisations in this area, such as ADULLACT, might prove to be the seed of such a centre.

When considering interaction with the “community”, it is important to realise that there is no single, homogeneous “Open Source community”. Rather, there are clusters of developers which form around projects, and there are many different, independent niches. A community does not simply happen; it must be built up and managed carefully. This may require management skills which are quite different from those necessary in a hierarchical environment.

It is important to understand the motives of the developers who contribute to free software projects. Contrary to a common myth, they do not see themselves as acting on primarily altruistic motivations. Rather, they usually perceive themselves as selfish and self-interested, and expect returns from their contribution, though not necessarily in monetary form. Nevertheless, roughly 50% of developers earn an income through their open source software work, and this accounts for a large majority of the most productive and experienced developers.

Very frequently, developers are also users of the software they are working on. They get involved because they need a certain functionality in a certain piece of software. For PA-specific software, it is likely that most of those who choose to join the community of a project will be associated with the public sector themselves.

Thus, a community for PA-specific software may include a handful of independent, volunteer developers – but is more likely to include developers employed at PAs, employees at firms that work (developing software or providing ICT services) for PAs, or freelance contractors who get paid by PAs or supporting firms. While such communities occasionally arise spontaneously, and are often self-organising once they start, they often need encouragement at inception. This is especially the case when the software at the core of the community is perceived to be “owned” by a single organisation, such as a specific PA.

Public administrations which use free software and release their own software under free licences offer numerous opportunities for the software and services industry. Unless they have a lot of in-house programming capacities, they will require firms to develop software for them under contracts. Companies are also needed to provide a unified point of contact for the maintenance of PA's IT systems, which are often quite complex; this will often involve service level agreements. On a more strategic level, public bodies may require consultancy services to assess their hardware and software situation, to prepare and execute software deployment or migration, and to integrate free software into mixed environments. These are only a few of the business opportunities arising from PA's use of free software.

Glossary

Apache HTTP Server
A free software/open source web server for Unix-like systems, Microsoft Windows, Novell NetWare and other operating systems. Apache is notable for playing a key role in the initial growth of the World Wide Web, and continues to be the most popular web server in use.

Debian
A widely used distribution of free software organized by the Debian Project.

Digital Rights Management (DRM)
Any of several technologies used by publishers (or copyright owners) to control access to and usage of digital data (such as software, music, movies) and hardware, handling usage restrictions associated with a specific instance of a digital work.

Firefox, Mozilla.
A free, open source, cross-platform, graphical web browser developed by the Mozilla Corporation and hundreds of volunteers.

FLOSS (Free/Libre Open Source Software)
The software is liberally licensed to grant the right of users to study, change, and improve its design through the availability of its source code and the right to distribute the changed program. FLOSS, free software and open source software are different terms to describe similar software, licenses, culture, and development models.

Free software

See FLOSS

Free Software Foundation (FSF)
FSF is a non-profit corporation founded in 1985 by Richard Stallman to support the free software movement ("free" as in "freedom"), and in particular the GNU project.

General Public License (GPL)
The GNU General Public License is a widely used free software license. The GPL grants the recipients of a computer program the following rights: the right to run the program for any desired purpose, the right to study how the program works, and modify it (access to the source code is a precondition for this), the right to redistribute copies, the right to improve the program, and release the improvements to the public (access to the source code is a precondition for this).

GNU
A free operating system consisting of a kernel, libraries, system utilities, compilers, and end-user applications. Its name is a recursive acronym for "GNU's Not Unix".

Kernel
The portion of an operating system which manages application software request for access to device drivers.

Linux (also known as GNU/Linux)
A Unix-like computer operating system. It is one of the most prominent examples of open source

development and free software; its underlying source code is available for anyone to use, modify, and redistribute freely.

GNU/Linux distribution
A GNU/Linux distribution is a version of a Unix-like operating system for computers comprising of the Linux kernel, all or part of the GNU operating system, and assorted libraries and applications. GNU/Linux distributions take a variety of forms from fully featured desktop and server operating systems to minimal environments for use in embedded systems or for booting from a floppy.

MySQL
A multithreaded, multi-user, standard query language (SQL) database management system (DBMS).

OpenOffice
OpenOffice.org is a free office suite application available for many different operating systems including Microsoft Windows, Linux, Solaris and Mac OS X. It is intended to be a compatible alternative to Microsoft Office. It supports the OpenDocument standard for data interchange.

Open Source Software
See FLOSS

Open Source Initiative (OSI)
A non-profit corporation dedicated to managing and promoting the Open Source Definition for the good of the community, specifically through the OSI Certified Open Source Software certification mark and program.

Operating system (OS)
A computer program that manages the hardware and software resources of a computer. Examples of operating systems for personal computers include Mac OS, Microsoft Windows, and GNU/Linux.

Project (or open source project)
A project is a temporary endeavor undertaken to create a unique software product.

Repository
A place where large amounts of source code are kept, either publicly or privately. They are often used by multideveloper projects to handle various versions and developers submitting various patches of code in an organized fashion.

Source code
Any series of statements written in some human-readable computer programming language. A computer program's source code is the collection of files that can be converted from human-readable form to an equivalent computer-executable form. The source code is either converted into an executable file by a compiler for a particular computer architecture, or executed on the fly from the human readable form with the aid of an interpreter.

Sourceforge
SourceForge.net is a centralized location for software developers to control and manage open

source software development, and acts as a source code repository.

Unix
A computer operating system originally developed in the 1960s and 1970s by a group of AT&T employees at Bell Labs. Today's Unix systems are split into various branches, developed over time by AT&T, as well as various commercial vendors and non-profit organizations.

References

Bayrischer Oberster Rechnungshof (2001): Jahresbericht 2001. München.

BearingPoint (2004):Server operating system Licensing & Support Cost Comparsion. Windows Server 2003, Red Hat Enterprise Linux 3 and Novell/SUSE linux 8. May 2004. Available online at: http://download.microsoft.com/download/7/3/9/739c7ab3-25c4-4b8c-9680-81ae10573b9d/BearingPoint.doc

Capgemini (2004): Online availability of public services: How is Europe progressing? Web based survey on electronic public services. Report of the fifth measurement. October 2004. Brussels: EC

Cowan, Robin. 1991. "Technological Competition and Variety: Issues of Diffusion and Intervention" in G. Bell, ed. Technology and Productivity: The Challenge for Policy, Paris: Organization for Economic Cooperation and Development, pp. 509-522.

Cowan, Robin. 1992. "High Technology and the Economics of Standardization" in M. Dierkes and U. Hoffman, eds. New Technology at the Outset: Social Forces in the Shaping of Technological Innovation Frankfurt/Main: Campus Verlag, pp. 279-300.

Cowan, Robin. 1995. "The Informatization of Government: From Choice of Technology to Economic Opportunity" Commissioned for the OECD/Tübitak Conference on "The Changing Role of Governments in Introducing New Information Technologies", Ankara, September 1993; in Science,Technology, and Industry Review, No. 16, pp. 195-223.

Cowan, Robin, 2002. "On the Number of Firms and the Amount of R&D" Economics Bulletin vol 12(6) pp. 1-9.

Cowan, Robin, Ferné, Georges and Foray, Dominique. 1991. Information Technology Standards: The Economic Dimension. Paris: Organization for Economic Cooperation and Development, [ICCP Report 25].

Cowan, Robin and van de Paal, Geert, 2000. European Innovation Policy in the Knowledge-Based Economy. Brussels: European Commission Directorate General: DG-Enterprise,.

Cybersource (2002): Linux vs. Windows total cost of ownership comparison.Available online at: http://www.cyber.com.au/cyber/about/linux_vs_windows_pricing_comparison.pdf

Deutscher Bundestag (2002): Deutschlands Wirtschaft in der Informationsgesellschaft. Berlin, February 2, 2001: Deutscher Bundestag. Available online at: http://dip.bundestag.de/btd/14/052/1405246.pdf

Dusollier, S., Laurent, P., and Schmitz, P-E. 2004. Open Source Licensing of software developed by The European Commission (applied to the CIRCA solution). European Commission DG ENTR. Available online at http://europa.eu.int/idabc/servlets/Doc?id=19296

eGovernment News (11 May 2004): Cost of open source software for government debated in

Ireland. Available online at:
http://europa.eu.int/ISPO/ida/jsps/index.jsp?fuseAction=showDocument-&documentID=2536&parent=chapter&preChapterID=0-140-194-349-360

Ettrich, Matthias (2004): Koordination und Kommunikation in Open Source Projekten. In: Gehring, Robert A. & Lutterbeck, Bernd (2004): Open Source Jahrbuch 2004. Zwischen Softwareentwicklung und Gesellschaftsmodell. Available online at:
http://www.opensourcejahrbuch.de/2004/OpenSourceJahrbuch2004.pdf

Evans, David S. and Reddy, Bernard J. 2003. "Government Preferences for Promoting Open-Source Software: A Solution in Search of a Problem".
9 Mich. Telecomm. Tech. L. Rev. 313, available at http://www.mttlr.org/volnine/evans.pdf

Frank, Roben (2005): E-government dans le secteur social:#développements futurs, 26 October 2005. available on-line at: http://www.ksz-bcss.fgov.be/documentation/ fr/documentation/Presse/ 60_ans_ONSS.ppt

Ghosh, R. A., (2003), "Copyleft and dual licensing for publicly funded software development", working paper, Available online at http://www.flossproject.org/papers/dual.htm

Ghosh, Rishab & Glott, Rüdiger (2003): Open standards and open source software in the Netherlands. A quantitative study on attitudes and usage in Dutch authorities on behalf of the office of the "Programma OSOSS", the Ministry of the Interior and Kingdom Relations, and the Ministry of Economic Affairs. November 24, 2003. Maastricht: MERIT. Online:
http://www.ososs.nl/article.jsp?article=8804

Ghosh, Rishab & Glott, Rüdiger (2005a): Open Standaarden en Open Source Software in Nederland. Een kwantitatief onderzoek naar houding en gedrag van de Nederlandse overheid in 2004 in opdracht van het programma OSOSS, het Ministerie van Binnenlandse Zaken en Koninkrijksrelaties en het Ministerie van Economische Zaken. Den Haag: Stichting ICTU / Programma OSOSS. Available online at: http://www.ososs.nl/article.jsp?article=17363

Ghosh, Rishab & Glott, Rüdiger (2005b): Free / Libre and Open Source Software – Policy Support (FLOSSPOLS): Results and policy paper from survey of government authorities. Maastricht: MERIT, University of Maastricht.

Ghosh, R. A. and Glott, R., 2005c. "The open source community as an environment for skills development and employment generation", Proceedings of the European Academy of Management Annual Conference, May 4-7, Munich. Available at: www.euram2005.de

Ghosh, R.A., Glott, R, Robles-Martinez, G. & Schmitz, P-E 2004. Guideline For Public Administrations On Partnering With Free Software Developers. European Commission DG ENTR. Available online at http://europa.eu.int/idabc/servlets/Doc?id=19295

Hermans, T., 2002. "Measurement of GFCF in software in the Belgian National Accounts". Joint ECE/Eurostat/OECD Meeting on National Accounts paper. Available at:
http://www.unece.org/stats/documents/ces/ac.68/2002/12.e.pdf

IDABC, 2005. Proceedings of workshop, IDABC@LinuxTag: Working with the Community: OSS in Public Administrations. June. Available at http://europa.eu.int/idabc/servlets/Doc?id=21209

Koordinierungs- und Beratungsstelle (KBSt) der Bundesregierung für Informationstechnik in der Bundesverwaltung im Bundesministerium des Innern (2003): Migrationsleitfaden. Leitfaden für die Migration der Basissoftwarekomponenten auf Server- und Arbeitsplatz-Systemen. Version 1.0. – Juli 2003. Berlin: Bundesministerium des Innern. English version is available online at: http://www.bmi.bund.de/Annex/de_25072/Download_englisch.pdf

Mangham, A. and Ghosh, R.A. 2005. London Borough of Camden: public procurement of open source software development. Published on EC DG ENTR/IDABC Open Source Observatory, January 25. Available at http://europa.eu.int/idabc/en/document/3804/470

Nagler, M. 2005. Open Source Adoption of the German Federal Office for Information Security. Published on EC DG ENTR/IDABC Open Source Observatory, July 22. Available at http://europa.eu.int/idabc/en/chapter/470

Oranje, C. van; Simmons, S.; Cave, J.; Weehuizen, R.; Glott, R.; Rothenberg, J.; Rubin, J. (2006): EUReGOV: Innovative and adaptive pan-European services for the citizens in 2010 and beyond. Inception Report. Prepared for the eGovernment Unit, DG Information Society and Media, European Commission. RAND Europe and MERIT. Available online at: http://www.euregov.eu/deliverables/reports/inception_report_v2.pdf

Parker, R. & Grimm, B. 2000, “Recognition of Business and Government Expenditures for Software as Investment: Methodology and Quantitative Impacts, 1959-98”, US Bureau of Economic Analysis paper available at http://www.bea.doc.gov/bea/about/software.pdf

Parker, R., Grimm, B. & Wasshausen, D. 2002. “Information Processing Equipment and Software in the National Accounts”. National Bureau of Economic Research conference paper, available at: http://www.nber.org/~confer/2002/criws02/wasshausen.pdf

Robert Francis Group (2002): Total cost of ownership for Linux web servers in the enterprise. Available online at: http://www-1.ibm.com/linux/RFG-LinuxTCO-vFINAL-Jul2002.pdf

US Department of Commerce, 2000. U.S. Computer Software Industry: Size, Firms, Establishments, Employment, Receipts. Available at: http://exportit.ita.doc.gov/ocbe/USIndust.nsf/806cbc35babba9838525695100784a38/538b5d24b61020898525696 2006c91c8!OpenDocument

Varian, Hal R. & Shapiro, Carl (2003): Linux Adoption in the Public Sector: An Economic Analysis. 01 December 2003. Berkeley. Available online at: http://www.sims.berkeley.edu/~hal/Papers/2004/linuxadoption-in-the-public-sector.pdf

Weber, Steven (2004):The Success of Open Source. Harvard University Press.

Wheeler, David A (2002): Why Open Source Software / Free Software (OSS/FS)? Look at the numbers! Available online at: http://www.dwheeler.com/oss_fs_why.html

Wichmann, Thorsten (2003): Use of Open Source Software in Firms and Public Institutions. Evidence from Germany, Sweden and UK. FLOSS final report – part 1. Available online at: http://www.infonomics.nl/FLOSS/report/

Winslow, Maria (2004): Evaluating the ROI of Open Source on the Desktop. Linux and OpenOffice are ready for your business. In: Linuxworld.com, February 18, 2004. Available online at: http://www.linuxworld.com/story/43720.htm?DE=