POOLING OPEN SOURCE SOFTWARE

An IDA Feasibility Study

Interchange of Data between Administrations

European Commission, DG Enterprise

Authors:

Patrice-Emmanuel SCHMITZ

Sébastien CASTIAUX

UNISYS

June 2002

© European Communities

Reproduction is authorised provided the source is acknowledged.

The IDA programme

IDA (http://europa.eu.int/ispo/ida) is a European Commission driven strategic initiative using advances in information and communication technology to support rapid electronic exchange of information between Member State administrations. The objective is to improve Community decision-making, facilitate operation of the internal market and accelerate policy implementation.

Its mission is to co-ordinate the establishment of trans-European telematic networks by:

Promoting implementation of sectored networks in priority areas
Developing network interoperability measures
Extending network benefits to EU industry and citizens
Co-operating with Member States authorities and Community services
Promoting convergence towards a common telematic interface.

If you would like to comment on this report or related issues, please e-mail to ida-central@cec.eu.int.

This report has been prepared under the sole responsibility of the contractor (Unisys Management Consulting Team). It does not necessarily reflect the view of the European Commission, nor does the Commission accept responsibility for the accuracy or completeness of information contained herein

For more information, please contact the author: Patrice-emmanuel.Schmitz@be.unisys.com

Table of Content

Management Summary.

Pooling Open source Software?

The Feasibility Study

The present Study is a feasibility study about pooling (or sharing, exchanging) software and knowledge between public sector administrations across Europe. A condition for greater re-use of software of the public sector is that software is available as open source. To encourage the sharing of software, the feasibility of creating a common software pool is considered. This would be a service combining a European knowledge centre and best practice. The legal, functional and technical constraints are evaluated within the present Study.

OSS Notion

Open source software is software where the author (the "licensor") gives some fundamental freedoms to the user (the "licensee"), inside a license agreement:

- 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.

What software, and why "Open source"?

Examples of open source software (OSS) are well known[1], but the aim of the study is not to focus on these "stars". The object of the study is the specialised software produced by the public authorities across Europe, to respond to the administration or more generally to eGovernment needs: administration of roads, hospitals and public health, education, tax payment and recovery, justice, territory management[2].

[1] The Linux operating system (or more exactly, the Linux kernel of the "GNU/Linux operating system") is the most famous OSS by far but is only the flagship of a numerous army of more or less illustrious representatives.
[2] This is a selection of best practice presented at the eGovernment Conference (Brussels, 29 November 2001)

So why “Open source” then? 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 purpose regarding license fees and the necessity to deliver the software with his 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.

The study purpose is therefore not of the advantages or disadvantages of open source and proprietary software. It is not to recommend any operating system or application, as the software developed for administrations run on multiple platforms (MS/ Windows, proprietary Unix, Linux etc.). It is not to take position in the commercial or sometimes ideological conflict between the advocates of free software distribution and the advocates of reinforcing intellectual and industrial property on software.
It is just to examine the pre-requisites and conditions (functional, legal, technical) of a pan-European pooling service.

The Ideas behind pooling

The ideas behind pooling software belong to three groups: Economy, Quality, Philosophy.

Economy

- When acquiring existing software for new applications (licensee point of view and interest): obtaining the best value for citizen’s money by reusing and adapting best practice software done by other administrations.
- Reducing costs related to maintenance (licensor point of view and interest): existing applications are costly and difficult to maintain and to develop according to new standards. “Giving” their software as open source to a developers community may perhaps provide maintenance and new versions for free…
- Sharing new software developments when no solution exists (common point of view): rather than to develop nearly identical solutions separately, why not adopt the open source development model to share the cost between a broader (trans-border) development team?

Quality

The sharing objective is not necessarily to spend less but to obtain a higher quality for the same amount of money:

- Speed up innovation by using funds to develop really new applications and not to re-invent parts that have been already developed by others.

- Allow countries to benefit from the advance of other countries: comparative studies showed “leading countries” having made significant advances in one or several specific domains.

Philosophy

- Promote collaboration between European administrations;
- Optimise the cooperation with the private sector, by concentrating investment on really innovative sectors and by promoting new support services;

The Study Summary

The Study is organized in six chapters, responding to six questions:

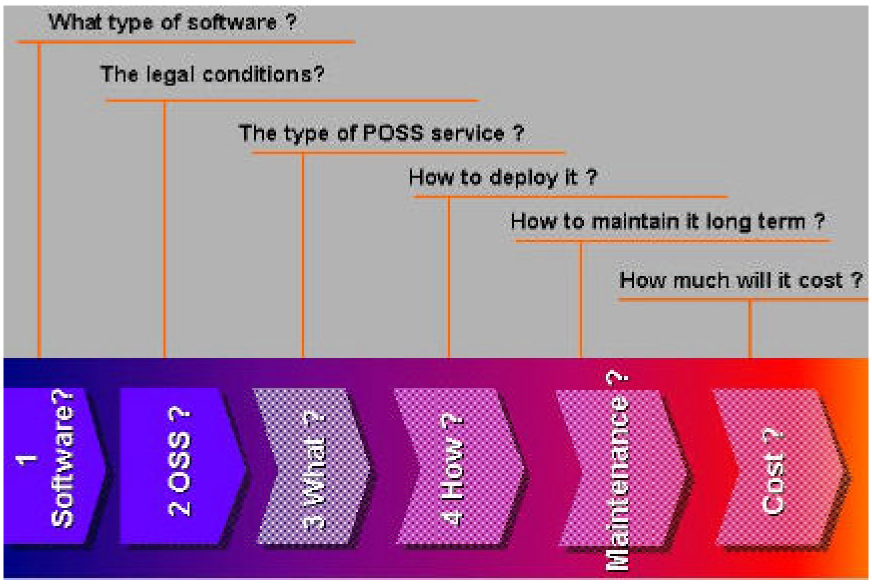


Figure 1 POSS study process

1. “Process and resource analysis”

The question “What type of software” is relevant, because 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? To answer these questions, the chapter analyses the processes of software development, acquisition and deployment, and the involved resources (human, organisations, projects, financial resources, distribution and information channels).

The main conclusion is that pooling existing software as open source 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;
- Documentation everywhere;
- A roadmap (navigate into the code as in a web site);
- A common trunk easily understandable + functional modules / no monolithic code;
- Software organised in many relatively small pieces of code (in order to facilitate individual ownership);
- Clearly identify (and declare) what parts are “mature/stable” and what parts are “to improve” (according to the “release often” principle);
- Launch 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 (new software) that will be recovered long term.

2. Legal framework analysis

Prior to share open source code with a community, the conditions of this sharing and the community itself must be defined: Other administrations? Everybody? Should the software be also delivered for commercial purpose to enterprises? The legal framework must therefore be analysed in order to define the roles, the responsibilities and borders between provider, user, intermediary service, and for guiding the choice of contractual conditions organising it: the licence and other provisions.

The conclusions are that the contractual framework is not limited to the software license, but includes:

- The general terms of the pooling service (what we call the POSS chart);
- The contract between the author and the POSS (what we call the mandate or “commission”, as the POSS will represent the author – licensor when contracting with the user - licensee);
- Specific agreements related to liability, competent judge and applicable law, patent issues;
- The license agreement itself that should be selected by the author – licensor (and accepted by the user – licensee).

Concerning the beneficiaries of the POSS service the common conclusion is that limiting the access to pooled software to the administrations only is not a major concern, however it should be limited to registered users (mainly public sector in a flexible sense). The limitation should not concern the end user itself and therefore must not be included in the license: it is not possible for legal reasons (we will see that, due to the "copyleft" effect, original OSS licenses as the GPL and MPL must be used without modifications in some conditions) and also for practical reasons: in an open source environment allowing re-distribution, the end user control is impossible.
Therefore, the limitation should be related to the POSS downloading service access and to the authentication needs related to valid contracting: only registered (European?) users (public administration, developers working for it, industry partners) should be registered to download. The possible sub-sequent redistribution by initial licensees will not be controlled (and will stay outside the POSS scope and liability if any).

Concerning the various open source licenses, they are all based on copyright and their effects, although estimated in conformity to the copyright conventions and the various European national applications, differ strongly.

With more than 30 different types or variants of OSS license available, the POSS should guide the candidate licensors with 3 options: the BSD, the GPL and a MPL variant[3].

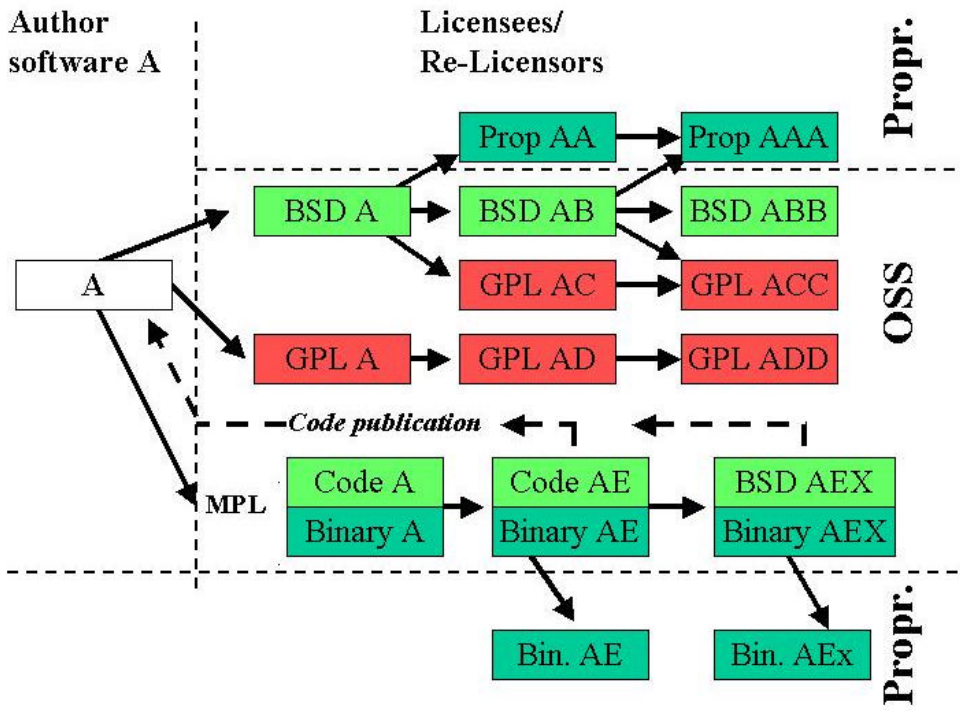


Figure 2 License generations

The BSD (Berkeley software distribution) license permits the widest panel of uses, especially when collaborating with the software industry: If software **A** is licensed by its author under BSD, it may become proprietary (distributed by

[3] Assuming that the licensor is free to select his license and is not already engaged by copyleft conditions, and knowing that the initial author is also free to use multiple licenses: for details, see chapter 2.

an industry vendor with a proprietary license) in its variants AA, AAA etc. At the same time another BSD licensee of the initial version A can improve it in AB and redistribute it under the same permissive BSD (variants AB, ABB etc…) or redistribute it under the conditions of any another OSS license (for example GPL for variants AC, ACC, etc.)

The GPL (General Public License) is the less permissive: all software **A** redistribution (of the same software, of improvements, of inclusion into a broader piece of software) must be done under the same GPL conditions (variants AC / ACC or AD / ADD in the above figure).
The GPL is less generally attractive for the software industry[4], but will gurantee a public authority that no version of its software will be "appropriated" in the future (assuming this point is important for a public authority).

The MPL variant provides an interesting compromise, where the code and the executable binary may be dissociated: the code will stay open (copyleft effect) and in addition the original author will be notified with all improvements (important to control and benefits from upgrades AE, AEX etc.). To the contrary, the binary object *may* be distributed with a proprietary license (prohibiting redistribution by simple end-user media duplication). In addition to the author interests, this protects commercial interests of possible distributors of derived works (they can both ask for a fair price and fight against piracy) and the interest of further developers: if they need to add technical improvements, they can access the code, modify it, recompile it and then redistribute a new " original" version (code and binary).

All the above indications will be provided for information only, as the choice of the license will be the sole responsibility of the licensor.

Last, the risk of overlap with private sector was examined, but due to the specific character of the concerned software (specifically developed by or for administrations), it was estimated that this risk is limited: In the POSS domain, open source applications and infrastructures will open a new market for service companies and solution providers, and facilitate quality improvement to European best practice. The global IT development budget and the part of this budget that will be outsourced to private sector developed are not questioned by the POSS.

3. Functional requirements analysis

Chapter 3 of the study was elaborated based on the result of our questionnaire: What type of POSS service should be expected by public sector administrations?
The main conclusions are as follows:

- Many projects developed could be used in other administrations (data exchange, groupware, human resource management system with a web interface) but today there is a lack of exchange means.

[4] Major proprietary software industry actors are lobbying against the use of the GPL, where other IT actors (more focused on hardware and services) accept it and use it in business.

- A European POSS portal should take into account similar sites that have already been developed at the national level, for example by providing links to their sites.
- A quick reference guide for the selection of the license model should be available on the site.
- Standards (coding, data format…) should be defined on the POSS portal and all the projects registered on the site must be compliant with them.
- Security aspects have to be carefully examined.
- The site must be a certain guarantee (a label) of the quality of the registered software, and therefore reflect user appreciations.
- Limiting the access of the POSS to administrations only is not a major concern.

To address these needs, the POSS should include a series of services:

- A multilingual portal with a clear site map, pan-European content and standard data format;
- Registered members management;
- Software submission, description, classification and downloading management…);
- Links to national initiatives and addition of any other link related to projects (software) and members;
- Library management to attach various documents related to software (technical documentations, user guides);
- News, Forums and mailing lists;
- Opinion surveys
- Software hosting for downloading (FTP) directly from the POSS
- A scientific committee. Even if no formal guarantee or quality audit will be possible, the conformity to formal quality criteria (documentation, manuals) and available test scenarios, as the advices of experts corresponds to a frequently expressed need.
- Legal advisory service, and contractual framework management (including the registration or log of all contracted downloads, by authenticated users, software, time stamps, agreed license types).

4. Technical design framework

In Chapter 4, the Study examines how the POSS should be constructed and deployed according to a technical framework of standards, together with illustrations of such possible frameworks (similar services and recommendations).

It includes the following topics:

- The definition of the technical framework of standards to respect when designing the POSS. This part presents the general site design and the security features.

- A list of possible tools requested to set up the POSS. This list is based on tools used by similar sites and that are proved to be compatible (operating system, web and FTP servers, database, e-Mail, statistics, link checker, scripting language).
  The tools presented here as ***examples*** are "**Open source**" (mainly because mature OSS tools are available and to avoid confusion and debate that should damage the credibility of the Study).
  However, it must be said that sharing public sector specific software thanks to open source licenses could as well be done – technically – by using one or more proprietary components if justified by consistency, global integration and delivery costs or performances.
- The different tasks to be fulfilled to set up the POSS, and for each task the identification of responsibilities
- Examples of similar existing tools with their components.

5. Maintenance and Interaction analysis

Making once a service or a portal is one thing, but the maintenance of it may present the most important part of a long-term investment.
Chapter 5 respond to the question "How to maintain the POSS ?" by identifying roles and defining the maintenance process.

The identified service roles are:

- System administrator
- Evaluator (of software)
- Controller (of daily POSS content, users, news, forums)
- Software (or project) responsible
- Developer
- User

The maintenance process include:

- **Components**
  POSS hardware and software such as the web server, the forum tool, the e-mail infrastructure, the link checker tool;
- **Content**
  maintenance of software index and home pages, links, news, and library indexes;
- **Services**
  member's registration, legal advice and framework, forums, mailing list, surveys, statistics.

For each maintenance process, the responsibilities are identified.
Next, the problem of the data integrity is described together with the solution proposed for the POSS site.
Finally, the possible contribution of organizations in Europe is evaluated.

6. Costs and financing options analysis

The last step in the feasibility study is a reflection on the costs: Prototyping costs (analyse and development), deployment costs requiring not only technical but also considerable informational efforts, maintenance and operation, periodic re-evaluation, upgrades and POSS application life cycle.

The first political choice is related to the POSS concept itself:

Why not starting at “no cost” like an open source software project, with one or more leading gurus and an army of free contributors, all of them working on a voluntary basis on spare time / free time to construct the POSS?

We answered negatively: the POSS is **mainly a service** that must be delivered constantly, and it is **not just a software** that volunteers can improve without constraints on their spare time. According to all OSS business models, there is no reason to invest less for a quality POSS service than for any other pan-European service. If this service is supported or even organized by the European Commission, the constraints concerning the quality, the number of languages, the availability and the neutrality will be high and must be delivered based on various service level agreements (SLAs) with professional organisations.

Therefore, the budget to foreseen is “normal” for this type of organization: an initial investment (prototype) between 510.000 and 1.060.000 euros, a deployment of 340.000 euros and 5 years of operations at 975.000 euro per year, bringing the TCO at about 6 millions on five years.

The return on investment will not be immediate, as the initial effort to evaluate and “adapt” existing software to open source pooling will be important and as new projects take time to gain their maturity. Therefore, starting from scratch, we estimate that a four to five year engagement is required to demonstrate (based on the POSS statistics and user inquiries) that the amounts of cost savings is higher than the investment, and who are the main beneficiaries. During this initial five year period, a public POSS funding is required. No funding by initial beneficiaries is realistic: their first role is to provide software. The large “Hardware and Services” IT industry is not interested, as their ambition is to concentrate efforts on specific applications (for example “Linux applications only to promote their server market”) that are not compatible with the POSS neutrality.

1. Processes and Resources Analysis

Objectives

The development life cycle of a software defines all the steps that need to be undertaken for the production of a new application. As such, the quality of the life cycle implemented will be the warrant of the quality of the software produced.

The objective of this chapter is to identify the processes and resources that should be involved in the OSS (open source software) development model, to guarantee the quality of the shared applications.

By “process”, we mean the logical sequence of operations to undertake in an OSS development or the identification of software that can become OSS with an appropriate license and their integration in a POSS (Pooling Open source Software) service.

Resources include

- human resources
- hardware and software resources
- funding
- organizations
- distribution and information resources

The processes

The processes in the development life cycle of an OSS are the following:

- *The software development*, i.e. the conception, design, programming, testing and implementation of the software.
- *The software acquisition*. The registration on the pooling site with the corresponding contract between the provider and the pooling site.
- *The software deployment*. The acquisition of the software by users with the corresponding contract between the users and the pooling site.

Software development

Developing a software only makes sense if this software responds to the needs of its future users. The software that will be registered on the POSS portal site have to respond to the needs of *several* administrations.

Once an administration has defined its requirements for a new software, it would consult the POSS site to see if its needs can be fulfilled by an OSS: either an ***existing software*** or a ***new development***.

For *existing software*, the main issues will be the legal aspects, the availability of documentation and the adaptability of the code.

On the opposite, for *software to be developed*, all the legal issues can be tackled from the beginning (rights and license). Documentation can be written knowing the pooling needs and development will take into account the need for the adaptability of the pooling software.

Existing software

For this category, the word "*development*" is not really appropriate (initially), since existing software only refers to applications already in production. Each administration has to ***identify its own valuable software*** that could be distributed as an OSS.

At least some preliminary fundamental questions should be investigated, and each of these questions corresponds to a key factor of success:

1) 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? Can we estimate (by preliminary personal contacts) the number of potential user and of candidate contributors?
2) Why should I provide "my software" as an open source to the community: Is it 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 *my* own problem: e.g., I cannot maintain my software anymore, it is too big or it costs too much, it is not well documented or initial developers disappeared and I just hope that by giving it as an OSS I will immediately benefit from free maintenance and upgrades?

3) Related to the question above, what will my future contribution be ? Am I (or developers in my administrations) ready to actively participate in maintenance, to distribute the work among interested contributors, to act as a “guru” having the best knowledge about it, to organize events and technical workshops around it in order to communicate and share a same level of knowledge and ownership with a new developers community, to support a web forum, a site with FAQ and technical information about releases, problem solved, tips etc.?

4) How can I estimate the maturity of my existing software: the stability of the solution in the production environment, the adequacy of the technical and functional documentation, the quality of functionality improvements to provide, at least in the next development cycle? 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.

5) Is my software code ready for open source sharing? This is one of the most sensitive questions, that may be illustrated by a case study comparison between Mozilla and KDE (out of many similar examples) :

> The Mozilla existing code (an Internet browser interesting a very large user community) was given as open source by its original developer (Netscape). The existing code that was discovered by the user community was largely unreadable, under-documented and incomplete (initially, it did not even compile correctly). The OSS project was originally staffed with Netscape developers, which had much better knowledge of the source than any external. In addition, the project’s schedule had no clear milestones and no incremental roadmap. Finally, there were no small code fragments, which could be handled by a single person (able to say "this is my code, my contribution, I have done this"). The project looked like a monolithic block of code, which could only run as a whole.
>
> Consequently the original Netscape funded developers out-coded and outwitted any volunteer programmers, making outside participation hard and ungratifying. The consequence of this initial situation was very hard to correct in later steps.
>
> Compare this with KDE (also written in C++, with a similar size and level of interest for a large community) which started from scratch as a spare time OSS project of an ever-growing group of volunteers. It released often, even unstable code provided it compiles and seems to run, with regular version numbers and press releases, even though the whole product wasn't stabilized. Each release contained sub-packages which are considered stable and others that are optional and clearly marked as experimental. The project is highly compartmentalized and many people "own" a piece of code, working together in small groups with a high degree of interaction.

The above case study (and what is said here about Mozilla also applies for other large “given” packages as OpenOffice) provides an example on the “how not to” and “how to” manage open source projects. The lesson can be applied to Public Sector pooled open source software too : the driving factor of any open source project is people and their interest in sharing the ownership of an evolving product.

In Europe, the request for clarity in documentation is complicated by the diversity of culture and languages: What if all existing comments are in French, in German, or in Swedish only?
The last questions and the above case study allow us to summarize the code-related prerequisites to share existing software:

- Documentation everywhere (in a common working language)
- Roadmap (navigate into the code as in a web site);
- Common trunk easily understandable + functional modules organised in many relatively small pieces of code / no monolithic code;
- Clear identification (and declaration) of the parts that are "mature/stable" and the parts that need to be improved ;
- Initial but evolutive repartition of "ownership / leadership" between diverse persons, from diverse organizations;
- Permanent discussion on requirements, objectives, priorities for further development

To state it clearly, the belief that any software may be given as open source, and then maintained and upgraded for free by an army of contributing voluntaries is an illusion[5].

6) After the above-mentioned evaluation process, the candidate provider must ***assess the adaptation costs***. Software is rarely developed with universal capabilities.
For example:

- Language issues must be taken into consideration. Is the software (menu, error messages, modular language related files…) multi or monolingual? If monolingual, what would the extension costs be?
- Some software have local specifications as tax rate, form composition…
- Some legal aspects are not valid for another country/region.
- Software must use standards for data formatting and data transmission in order to remain compatible with the new products on the market. These standards must be available on the POSS to check against the code.
- Documentation (technical documentation, user and administration manuals) and comments in the code must be comprehensive and understandable (clear naming convention, UML standard description).

The provider can then ***adapt the code*** to better match the POSS standards and to increase the flexibility of the software, or at least have a clear vision of the roadmap to adapt it.

[5] *Several persons we contacted were questioning the reusability of existing software.*
Patrick Gendre *(*CERTU/dept Systèmes/groupe Gestion du Trafic et Télématique FRANCE*) wrote on the ATICA forum dedicated to this study:*
"… in practice, reusability is not easy. Beyond legal aspects, it requires particular organisation demanding investments hard to carry out. Sharing software demands strong implication as well from the providers as from the users who have not always the competence and availability for that."
Hubert Tournier *(*Deloitte & Touch ERS/Secure e-Business, participant to "Atelier du libre" dedicated to this study in Paris on 14/02/2002*): "… Few people have the competence to adapt code to their needs, even among the computer scientists…"*

7) In parallel, and for each software, the ***rights have to be checked.*** The provider must be sure that he owns the rights allowing him to distribute the software, particularly if it was developed by partner companies.

8) Then, the provider must be prepared to ***choose an appropriate license***. For this purpose, the pooling site should include a quick reference guide for the selection of a license (see chapter 2), according to the provider's strategy (e.g. if the provider allows or not the use of his OSS code within a commercial product which is sold as a proprietary product).

9) When legal issues are tackled and adaptation costs are known, there is a need to ***provide an adequate user documentation***, at least a user and an administration manual together with the technical description of the product. The minimum requirements for documentation will be available on the pooling site. The European Commission could decide to make the documentation available in one or 2 languages only, which would also be defined in the standards of the POSS documentation.

The software is now ready for the next step: it can be handed over to the POSS evaluators and follow the acceptance procedure of the POSS.

New software

If a software is developed after the creation of the pool, the legal issues can be tackled from the beginning e.g. if partner companies develop the software, the contract must specify the rights for redistribution.

A new distinction can be made between software that will be developed within an administration and put on the pooling site when finished, and software that will be developed following the open source development model.

The open source development model implies sharing the work between people who are geographically distant. It requires collaborative work through the Internet, as opposed to a development executed within an administration or by a partner company. Therefore, it is important to clearly define the role and responsibilities of each person, clearly describing the interfaces between the application modules and giving the requirements per module. It is obvious that such developments must take into account the standards defined on the POSS Portal site.
We think that this kind of distributed development must not be taken into consideration for the first POSS implementation. The first objective should be to put the service in place with software developed internally by the administrations. However, a cooperative development must be possible in the future and the tools proposed in chapter 4 provide the components for this kind of development.

Developing from scratch according to the open source model is not without budget impact for the original author: In general, the initial costs are higher (about twice higher) due to the requested quality level and respect of standards implementation. This investment will be recovered later (with a facilitated support and maintenance, with contribution of a community).

In addition, the fact trying to develop for multiple platforms may again increase seriously (by 3) the development cost

Each OSS development project should be registered on the POSS site receiving a “Proposition” status as soon as its development is considered. This way, users are informed of ongoing decisions or developments and they can participate or give their opinions and requirements.
The software development model includes the following steps:

- **Requirements analysis**: before starting the software development an overview of the system and its desired functionalities must be provided. The specification and requirements documents are prepared. The specification also sets the boundaries of the projected system development. This does not mean that the system cannot be further expanded during the development cycle (change management), or after project completion (future enhancements).
- **Preliminary design**: this phase will refine the documents of the previous step to produce an accurate description of the system. The modules and the interactions between modules are defined. This is the most important phase of the software development cycle. Changes at this stage are less costly, and promote a better-designed system than changes made later in the development cycle.
- **Detailed design**: each module identified in the previous phase is described in detail. At this level, each module must have its interface completely defined and the unit test plans must be ready.
- **Implementation**: the modules defined have to be coded. A module is finished when it has been coded, tested and used successfully by another module. Tests are executed in accordance with their description in the detailed design.
- **Integration**: when all the modules are available, integration at system level can begin. All the modules are put together as one piece of code, compiled and linked to build one package constituting the system. Developers bring basic system level tests during integration.
- **Test**: after integration, the testing team begins to work to ensure the system meets the specifications and the defined requirements. Issues are addressed and fixed by developers. Once the system has been delivered, users acceptance tests are performed. Any issues that are raised by users during this time will be addressed and fixed or further enhancements are made to the system. The final system will include the issues raised during this testing phase.
- **Installation**: after testing and acceptance, software can be put in production. This stage includes writing installation scripts and packaging.
- **Maintenance**: maintenance involves the whole of the operations required to maintain, within prescribed limits, any element of the software.
- **User training/software documentation**: after the delivery of a new software, some user training or software documentation may be required. This will depend on the number of users and the complexity of the software developed. For POSS developments; documentation must be written taking into consideration the minimum requirements defined on the pooling site (technical documentation plus user and administration manuals).

The open source development model can include all the above-mentioned steps, but it is rarely the case. Some phases are not completely executed as described and others are merely skipped because software development involves a lot of resources that cannot be afforded by an open source community. Usually, open source development begins when an individual feels the need for some software functionalities. He begins to develop and publishes his code on the Internet. Other people can then bring their contribution/comments to the project.

- **Requirements analysis**: usually, open source developers produce software they need or they wish. If, after several versions, their software reaches the critical mass and becomes stable, it will be distributed as an archive file through Internet and users can use it and add functionalities. A specification document is built through diffusion list or discussion forums where users and developers talk directly together. This can be a very rich process but it can also lead to conflict due to the large amount of actors.
- **Preliminary design**: this step is often skipped in an open source development. Usually, the actual system architecture appears with the second or third software version but no document reflecting it is written. This constitutes one of the main issues of the open source developments.
- **Detailed design**: as no preliminary design really exists, it is difficult to produce a detailed design document. Except for some important projects, the detailed design is also skipped. It is really an issue since the lack of architecture documentation prevents the use and maintenance of valuable pieces of code.
- **Implementation**: for open source developers, the implementation is really the funny part of the work. Coding is their first motivation. This leads generally to "clean", suitable code. The fact that open source is aimed at large public leads to standardized software usable on several platforms. The main difference with usual software development is the code review that is not formal. Versioning is also difficult but free tools are available (CVS[6]).
- **Integration**: integration consists usually in the following steps : writing short user and administration manuals, ensuring that the software can be deployed on most common platforms, cleaning the "makefile", making an archive, uploading software on an ftp server and posting letters on forums or diffusion lists. For most open source projects, there is no integration test.
- **Installation**: developers write installation scripts before putting the software at disposal. Installation consists of downloading the archive file and following the installation instructions.
- **Test**: open source software benefits from the best tests due to the large community of users. Ten or hundred other developers track down the bugs by reading the code instead of merely executing it which is a big advantage of open source software.

[6] http://www.cvshome.org/

- **Maintenance**: even if problems can be submitted in forums or thru distribution lists to a large community trying to solve them, there is no guaranteed maintenance for the OSS products. Maintenance is hard to plan because of different co-existing variants. But, on the other side, it is an opportunity for service providers to propose assistance and maintenance contracts and so promote open source software (Minoru[7]).
- **User training/software documentation**: user training supposes a lot of resources that cannot be afforded by the open source model, as does software documentation which is generally reduced to the minimum.

Figure 3 shows the software development process.

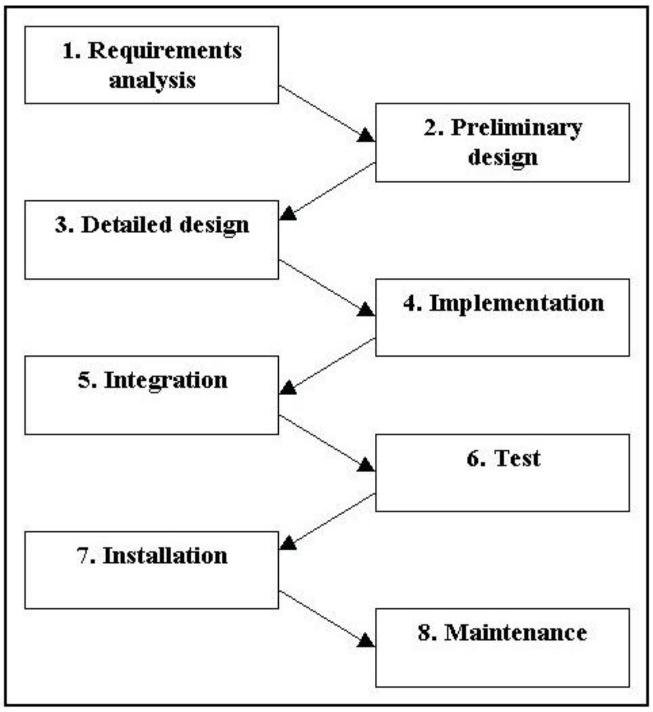


Figure 3 Software development process

The role of the POSS in this software development phase is:

- to provide the standards (code, documentation…) which will be used for all the phases of the development life cycle (e.g. naming conventions, UML);
- to provide an index of ongoing projects in order to avoid effort dispersion;
- to provide guidelines for assessing adaptation costs (for existing software);
- to provide a quick reference guide for the selection of a license;
- to identify all the people involved in the project;
- to facilitate cooperation and to increase a trans-border European developer’s Community. For example, the testing phase of a web-oriented project can be made throughout Europe with increased efficiency

[7] http://www.minoru-development.com/

Software acquisition (the contracting process)

When a software is ready for production, it will be submitted to the POSS portal site. It will first go through an acceptance test. The POSS experts must ***evaluate the software*** in terms of:

- ***Opportunity****:* experts must determine if the software is interesting for other administrations and if there is a need for such a software.
- ***Quality***: code and documentation quality must be evaluated.
  The code must respect standards, include sufficient comments and be easily adaptable (configuration files, no hard-coded parameters…).
  The documentation must include at least the technical description, a user manual and an administration manual.

This phase of "*Quality Assurance*" is very important because it guarantees the quality and the reliability of the software made available on the pooling site. If a lack of quality appears, the software is returned to the provider for improvements. For this purpose, a "quality form" must be defined to facilitate the communication of the software weak points. If the software passes the acceptance tests, it is registered on the POSS site.

The contracting process implies two parties:

- the provider
- the pooling site.

The types of contract available are defined in chapter 2.
The provider must accept the contract and fill out a specification form describing his software. The contents of the software form are described in details in chapter 3. It contains, among others, a software description, the license type, a short description of the available documentation, the support provided and the usage limitations. The registration of this information can be simplified by multiple-choice options.
A software responsible is also registered together with software developers (for small projects, there can be only one person who is the software responsible and the software developer). Users can contact the software developers later to get support.

If the acceptance testing process is too expensive, another approach can be chosen, where everybody is allowed to register software on the POSS portal site without passing any quality assessment. Nevertheless, an alternate process must be put in place to ensure a minimum quality, e.g. a software deletion policy based on the download rate, the average rating from the users, the update frequency …

If the provider has an FTP server at its disposal, the pooling site will provide a link to it, enabling users to easily connect to it and download the software. Otherwise, the files have to be uploaded on the pooling server. Space for the project home page can also be provided on the POSS web server.

Figure 4 shows the submission process.

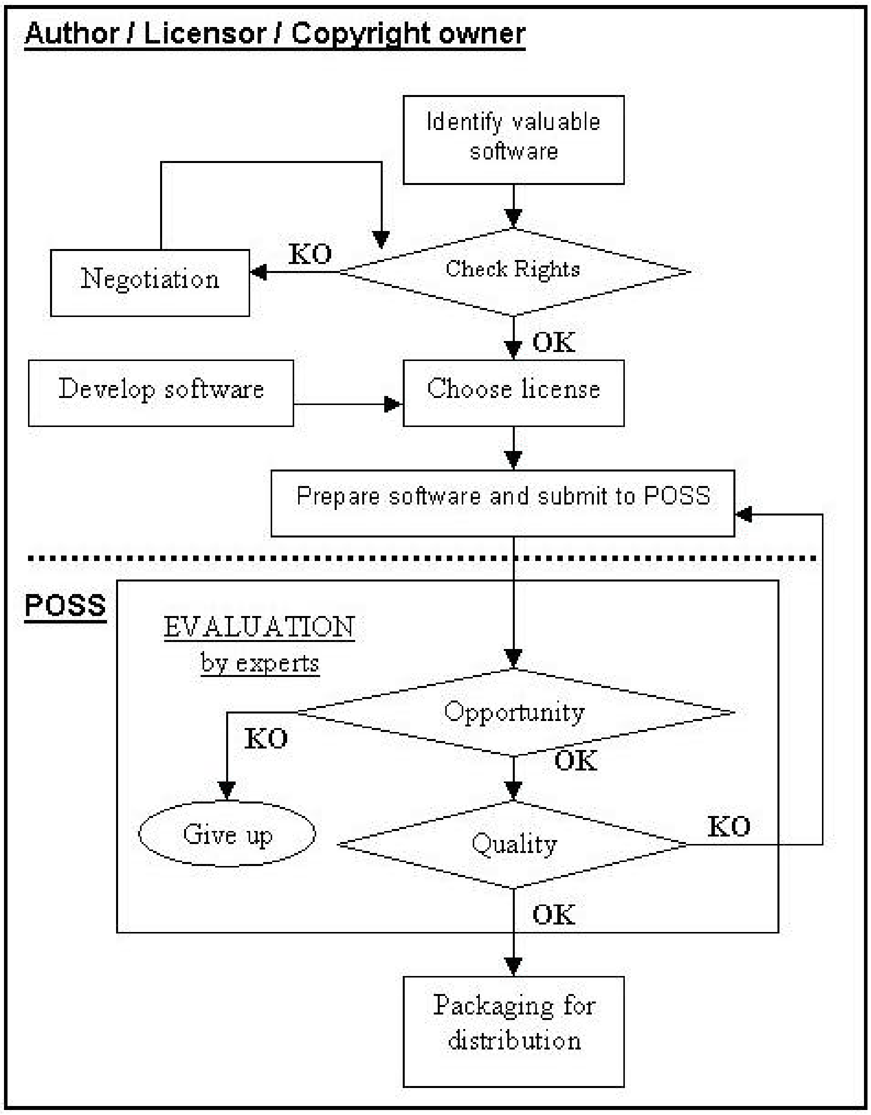

Author / Licensor / Copyright owner
Identify valuable software
Check Rights
KO
Negotiation
OK
Develop software
Choose license
Prepare software and submit to POSS
POSS
EVALUATION by experts
Opportunity
KO
OK
Give up
Quality
KO
OK
Packaging for distribution

Figure 4 Submission process

Software deployment

When a user has identified a need for a software, the first step he will take is the search in the POSS software index.
Software can be located on basis of the contents of its software form (software category, supported operating system, software language…). The searching tool also allows users to locate software items according to several criteria.

If several software items meet the user's requirements, a choice must be done. To this purpose, additional information can be asked from the software responsible and software developers registered on the POSS. This information could also be asked on forums and through mailing list.
A software users directory could also be available to allow future users to contact people who have already experienced the software.

When the software has been chosen, it must be downloaded. A second contracting process implies the pooling site and the user who wants to acquire software from the site. Future users have to accept the contract terms before getting the software.
Figure shows the contracting links. Legal aspects (license, contract…) are covered in more details in chapter 2.

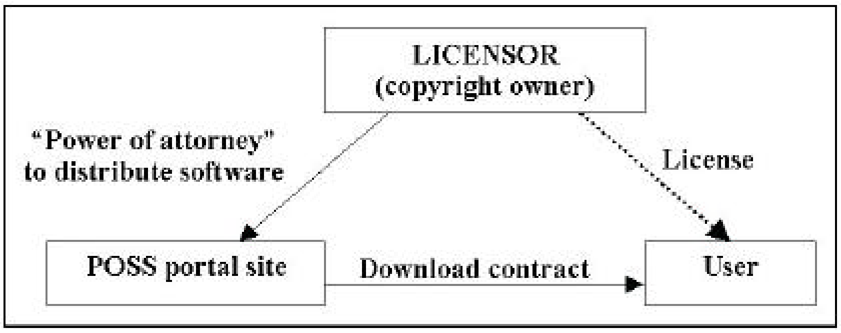


Figure 5 Contracting links

The software includes:

- Binary software distribution
- Software source code
- Code documentation (comment)
- Technical documentation
- User manual
- Administrator manual

For the set up and the maintenance on the user environment, support can be asked depending on the provider's conditions.
Users can also get support through forums or mailing lists dedicated to the software or by asking questions directly to the software developers registered on the POSS.

Figure shows the deployment process.

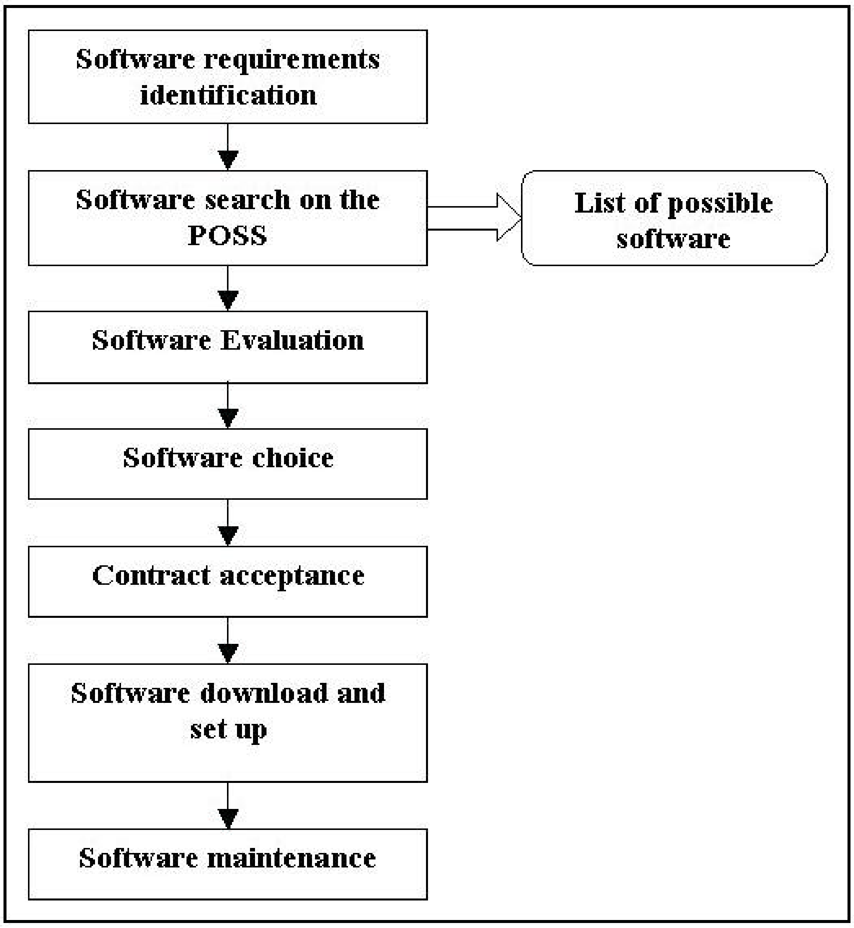

Software requirements identification
Software search on the POSS
List of possible software
Software Evaluation
Software choice
Contract acceptance
Software download and set up
Software maintenance

Figure 6 Deployment process

The resources

Human resources

Several profiles will be needed to set up and maintain the POSS portal site:

- ***System administrators***. The system administrators will be responsible of the administration (hardware and software) and the operations of the POSS system. They must have a good knowledge of the technologies used (mainly open source) to build the portal site.
- ***Software evaluators***. The software evaluator will be responsible of the quality of the software registered on the POSS portal site. Their responsibilities will be to accept or to reject submitted software. They must have a strong background in software quality assurance.
- ***Site controllers.*** They constitute the "*steering committee*" of the POSS portal site. They will be responsible for
  - the definition and the maintenance of the standards used on the POSS
  - the contents of the POSS (links, news and library)
  - the public forums moderation
- ***Law experts.*** The legal framework of the software distribution has a continuous evolution (new software patents, new licenses…). Law experts will be responsible for following the legal evolution of the software distribution and advising the POSS steering committee in this domain.
- ***Site advertisers***. The POSS portal site must be recognized as a reference among the European administrations. The site advertisers will be responsible to make the POSS portal site known and to keep users in touch with the site evolution. This can be made through newsletters, documentation, meetings, and seminars…
- ***Translators***. As the POSS must ensure multilingual diversity, translators will be needed for the software documentation, and maybe for some user interfaces.

Software

All software will be registered on the POSS portal site and will be accessible through a software index.
Registering the software as soon as they are proposed for development is a way to inform users about their existence and to give the status and maturity of all ongoing OSS processes inside the administrations.

For each software, a software identification form must be defined, as proposed in the previous study made by Unisys for IDA[8]. We suggest to keep the proposed form for the pooling site (see chapter 3 for more information).
The portal site will impose a common language for the comments in the code and standards for the coding styles (naming variable…) in order to develop common work practices throughout Europe. Some examples can be found at

[8] *IDA study into the use of Open source Software in the Public Sector (http://europa.eu.int/ISPO/ida/)*

- http://www.freebsd.org/doc/en_US.ISO8859-1/books/developers-handbook/index.html
- http://www.shmoo.com/securecode/

Audits can be set up for coding and security.
Existing sites (Sourceforge.net[9], Euspirit.org[10]…) can be evaluated to find relevant software for the POSS portal site. A remaining problem is that the language used in such sites is generally only English.
Although those sites provide a lot of interesting information, they raise the issue of the selection of relevant software. A lot of software are said to be "natural" (no identifiable authors), "orphan" (no more active authors) or "dead" (no more active developers). *One of the roles of the European portal* would be to provide users with pointers to relevant software only.

Funding Open source projects

The Open source business model and the costs and funding of the POSS service itself will be examined in chapter 6, but what about a POSS contribution to the general funding of open source projects?
Typically, the first free software development (especially in administrations or universities) does not start with initial funding: it starts with a small group of persons who try to solve a specific problem, or who want to implement an idea of theirs. Then they publish the software on the Net, others learn about it, find it useful or interesting, and develop new features which are needed for their own applications.

As it was noted by Thomas Roessler and Kristian Köhntopp [11], a large portion of the free software development currently going on is still done by volunteers in their spare time, often beating commercial software in efficiency, stability and quality when the following three conditions are met :

- Extremely low international transaction and publication costs;
- Enough educated persons able to create software (the community) ;
- Free time available, and motivation to spend this time on free software.

However, certain critical factors can inflate the costs beyond the budget of free developers.

A) When growing in importance and sensitivity, the project and the number of contributors (version control system) may require powerful dedicated machines, software and human resources for operations. As the software of the POSS would be developed on various Public Sector infrastructures and as other "software factories" exist, we do not see a direct need for the POSS to provide such an infrastructure.

[9] *http://sourceforge.net*
[10] *http://www.euspirit.org*
[11] Roessler and Koentopp, "*Hoe free software development can be supported*" http://www.koehntopp.de/kris/artikel/oss_funding/

B) The legal framework costs (e.g. intensive patent search before publishing[12]) can be heavy.
In this field, the POSS role may be important, since contributors are usually not aware of all the possible issues. A dedicated legal advice concerning license issues and an assistance in patent investigation is one of the POSS requirements.

C) Launching (marketing) or making business from OSS development may rapidly consume all the time from leading project actors and require expensive travel costs. Key developers may then start to step back from active development because of lack of time and resources. This may justify the need for external funding.
Funding possibilities already exist (provided by some private industries and by European and National governments as, e.g., the European IST 5th framework programme). Therefore it should not be a priority for the POSS itself but it could play an informal peer-to-peer coordination role (forum) and report/link to national and European funding possibilities.

Organizations

Various organizations or public authorities can support the sharing of software development. For the first implementation of the POSS portal site, we advise to limit the service to administrations only. It seems very difficult to get support from a hypothetical developers' community throughout Europe.

The organisations involved in the POSS processes are:

- ***European administrations***. The POSS portal site will be set up for European administrations and they must be closely involved in the processes. They will be the main software providers and users.
- ***National agencies***. National Agencies have an important role to play inside or in connection to the POSS portal site. They have strong knowledge of the administrations needs and have already contacts with the administrations at the national level. They can acts as links between the administrations. A typical example is the *ATICA* in France. The main objectives of the Agency for Information and Communication Technologies into Administrations are:
    - the definition of a general framework of interoperability between administrative information systems
    - the recruitment of computer scientists,
    - the regular diffusion of information both to the administrations and to the public.
- ***Companies***. Administrations often ordered software from private companies. Those companies can be contacted to adapt this software to the requirements of other administrations. This will imply lower development costs than brand new software. Companies could also be contacted to provide trainings or support for software (several companies like Minoru[13] are specialised in open source software adaptation).

[12] See chapter 2
[13] http://www.minoru-development.com

Distribution and information resources

Several means are available to ***disseminate information*** about the POSS portal site:

- The site itself, registered on web search engines.
- Links from other sites of the Commission and National Administrations
- Mailing lists from the POSS, the EC, the National Administrations, Developers Associations
- Meeting and conferences
- The Information Points of the EC
- Newspapers and magazines
- White Papers
- CD-ROM's

Those supports must be evaluated and the most pertinent ones must be intensively used to make the POSS portal site known and to keep users in touch with the site evolution.

For the ***software distribution***, Web and FTP servers will be needed. However, the usage of FTP servers must be restricted. It's a loss of space and time to store the same information at several places. As often as possible, pooling sites will only provide links to other existing sites.

2. Legal Framework analysis

Foreword

Although it may disappoint many IT enthusiasts and free developers, here is the reality: *The core of open source software is* ***not the code****. It is the* ***license****.*

This is sometimes hard to accept[14]. This is also the reason why the legal framework analysis is so important. We will examine various aspects of the subject:

- Terminology
- Copyright
- Patents
- Trade Marks
- Fair competition
- Licenses content
- Contract validity / license acquisition
- Liability

The purpose of the lawyer is not to establish or to protect what is "good" against the "bad" in general, not even to define ethical conduct for organisations and citizens, or to achieve a "justice ideal". It is to organise society activities according to a legal certainty. "Good" is achieved when the rules of the game seems clear for everybody and are applied the same way by every judge. "Bad" is uncertainty.

Open source software is at the intersection of many disciplines and contradictory interests: IT sciences (various domains) and law (various domains also as industrial/intellectual property, contracts and liability), politics and eGovernment (constitutional freedom to access and use information, freedom of expression), economics, competition (various business models), and philosophy (altruism, liberalism / free enterprise, property).

Parliaments have voted so far no specific rules covering the OSS matters, and therefore in case of litigation, the judges will have to apply general principles, and the result may vary form case to case.

In addition, open source software, its development model, and its distribution model are trans-border by nature. It would not even exist without the most used and shared open source reality: the Internet. This trans border-character allows software to be developed, loaded, downloaded and used everywhere in the world, without many customs, contractual or administrative restrictions.

[14] In many seminars, symposiums, colloquiums etc., it is often noticed tenacious traces of animosity of IT specialists against lawyers in general: perceiving these non-IT people as threatening developers freedom and impeaching them to deliver their Art.

Opposed to this, law is still territorial. Laws are made according to countries' national procedures and even international treaties have to be translated in national laws and practices that are different.
Law system families themselves are different, starting with the famous distinction between on the one hand "common law system" (U.K., USA) that are mostly based on case law and on the other hand "civil law systems" (continental Europe) that are mostly based on written law compilations.
In this trans-border context, with many possible rule variants applied by many possible judges from many possible countries, certainty can only be approached, never totally reached.

Open source software and the Internet are not "outlaw zones" and contract laws, copyright protection laws etc. fully apply to new technologies. New technologies may require an evolution or a re-formulation of the law, like for e-Commerce or electronic signature, but the same principles exist and apply: the valid formation of contracts, the protection of intellectual property (copyright or patent), the procedures to fight against crime (fraud, child pornography) existed before the Internet. The two real legal issue of the Internet world are:

- Virtual character: as the subject is software and more generally information, it can be transmitted and duplicated in great number, without owner's eviction, but without author's control. The virtual character requires revisiting law principles in a new reflection on direct or circumstantial evidences.
- Trans-border character: the problem is not to whether there is a law. The problem is to know which law is applicable. In this field, the Internet reinforces the utility of the existing International Private Law rules.

If a situation appears to be quite certain at a specific time, life will change it. Laws are moving, evolving under contradictory society and economic pressures. What is true today may not be true tomorrow, and some practices are in contradiction with the law.

A typical example is software "patentability". This is highly relevant open source software, as it is rightly perceived as the main threat against programming freedom.

a. According to the well-established European law (the European Patent Convention art. 52, 2c) programmes for computers are not regarded as inventions and therefore cannot be patented.
b. Despite this "apparently clear and obvious" wording, the European Patent Office has – over the years - widely followed the American practise of large patentability and has massively registered thousands of software patents, demonstrating the "theoretical" value of the law when a dominant actor interprets it differently.
c. Reacting, the European Commission (Directive proposal February 20, 2002) has launched the concept of "computer-implemented invention" that can be "realised wholly or partly by means of a computer programme" at the condition that it is "susceptible of industrial application, and makes a not-obvious technical contribution to the state of the art". (computer programme vendors that are interested will produce papers explaining that their software respond to the conditions…)

d. The future of the directive proposal is still uncertain as discordant reactions are expressed at the European Parliament and within Member States.

The above process clearly demonstrates the temporal, uncertain character of the rule. Law and even more case law offer many other similar examples.

The paradox of lawyers' motivation for changing or establishing new rules is their permanent quest for certainty. This was one of the strongest motivations of the Commission, when it noticed that until then the majority of national level jurisprudence in the field of computer-implemented inventions was developed in only two Member States: Germany and the U.K. and that even these two have decided differently on important questions as the definition of patentable matter. "*This suggests strongly that the courts of other Member States, in the absence of any harmonising measures, could well come to widely diverging positions if and when confronted with cases to decide in this field*"[15].

Our public sector authorities are involved daily in a massive amount of litigation (from tax payer or town planning contending parties to victims of natural diseases). When making the list of risks that governments have to address , we have – to be honest – to conclude that in the face of all potential risks, distributing public sector software is probably not the most dangerous of all.

After many years of existence the GPL license for example – that is used in the most important number of OSS projects - has still never been really tested on court[16]. One reason is the reduced number of conflicts related to open source licensing. Another reason is the fact most potential litigations are solved by amicable agreement out of court ("a l'amiable"). This is due to the nature of potential damages: compared to definitive physical damages (injuries, destructions, death) the simple fact of stopping software distribution, compromising, changing a name or paying a license fee is generally enough to avoid long and costly litigations.

[15] 20 February 2002 Directive proposal memorandum

[16] After years of existence, the FSF declares having enforced the GPL in several occasions, but "in a confidential setting without the need of court action". The first action is pending (February 2002) in United States District Court in Massachusetts: "Progress Software Corporation vs. MySQL AB". Progress lost the right to distibute MySQL due to a violation of the GNU GPL. Progress distributed a proprietary software component, Gemini, that was combined with the GPL'ed MySQL database system. Gemini is said to be linked statically with the MySQL system to form a single binary program. The close linking of the two programs (GPLed and proprietary) in a single product makes that – according to the copyleft principle, and if established in court – the whole product should also be GPLed.

An other example of "extra-judicial setting" was found by the FSF "Finite State Machine Labs Inc." (FSMLabs) discussions concerning the Open RTLinux patent license, covering patented components. After discussion and proposing a version 2 of its license, FSMLabs became compliant with the GPL. The V2 Open Patent License grants the right to use U.S. Patent No. 5,995,745 in GPL-covered free software without payment of a royalty, and protects GPL use of the RTLinux process.

As reported by Eben Moglen[17], when the GPL is violated the first step is a report, usually received by email from <license-violation@gnu.org>. This stage was reached dozens of times a year. A quiet initial contact is usually sufficient to resolve the problem. Parties thought they were complying with GPL, and are pleased to follow advice on the correction of an error.
In rare situations where the scale of the violation or its persistence in time makes mere voluntary compliance insufficient, the FSF works with organizations to establish GPL-compliance programmes within their enterprises, but in approximately a decade of enforcing the GPL, the payment of damages for violation of the license was never asked for.

These considerations should of course not temper the care of European public authorities to check and establish carefully the validity of their rights and to avoid law infringements (not only concerning copyrights, but also concerning patent and competition) by case impact study and an appropriate contractual framework.
Indeed (compared to most "de facto" open source communities), the governments and public sector in general fulfil two conditions:

- They are "visible" clearly identifiable as legal bodies
- They are financially solvent

Therefore, the risk exists to jump from a "no case law" to a "to much litigation" situation…

Clarity in legal rights and obligations is a secure way to restraint risks: perhaps not direct damage risks if any, but mostly indirect damages like loss of business, time, and confidence due to improvisation and to the fact that " what everyone has to do and can do" is left uncertain.
For these reason, the reports between:

- the licensor (original author or more generally "copyright owner" as the administration is regarding programmes developed by its employees and contractors)
- the distributor (and this includes the POSS service)
- the user

must be clearly defined, in the license itself and within a framework of other contracts, as the software license (and this is the case if a standard text like the GPL is chosen by the licensor) do not cover all requirements (as clear indication of applicable law, competent judge etc.).

[17] Professor of law and legal history at Columbia University Law School, General Counsel of the Free Software Foundation – see http://www.gnu.org/philosophy/enforcing-gpl.html

The Terminology

It may often be confusing to distinguish between terms as “shareware”, “freeware”, “public domain software” “open source software” “free software”, commercial and proprietary software.

Freeware is a quite vague term that should be avoided here: It is usually used when a piece of software is given at no cost, even if the programmes are released only as executables, with their source code not available. For example, you can download the Adobe Acrobat Reader as a freeware, but the software is still proprietary. The same remark applies to all terms related to the price of the software as “give away software” or “sell software”[18]

The term **Shareware** also covers a sales concept, where software is usually distributed free of charge for a limited period of time or for a limited use, mainly to give the user the opportunity to test it before buying it[19].

Similarly, the term **Commercial software** will not be used here.
A commercial programme can be free or non-free, depending on its license. A programme is commercial if it is developed as a business activity, and it may be the case for free or open source software. The Linux packaged distributions, for example, are commercial.

Public domain software is software for which copyrights do not exist . Although this notion is invalid in Europe (but can be understood in US law) it is often used for software any one can use for any purpose, without any restriction. In addition, the availability of the source code is not granted

Proprietary software
The real contrary of Free software is “Proprietary software”, where in general the user has only limited rights to use a product, on a specific machine, sometimes with a specific power or processor, sometimes with a limited number of signed or concurrent users, or related to a limited amount of material

Example:
Software X is used to design and use databases containing clients’ records.
It is delivered with a key enabling to operate:

[18] Another variant is **Bundled Software:** provided with or added on to commercial software at no extra charge, but usually under a very restrictive license and without freely available source code; e.g. Microsoft Internet Explorer

[19] In addition to Shareware, often developed by a single person, usually unsupported and without published source code, with the hope of receiving monetary donations from users, software may also be distributed as:

- **Demos/"Crippleware":** no-cost or low-cost subset of a commercial software product, without freely available source code, without support, and usually lacking key features

- **"Consortium-Ware":** source code shared among a group of companies, but not freely available to the public; e.g. Motif

- **Non-commercial use only":** software which is typically downloadable for free from the Internet, maybe even with source code, but which has serious licensing restrictions which limit its use; e.g. Netscape's browsers prior to January 1998

- with your specific owner name;
- on machine number nnnnn;
- with a processor speed of zzzzz;
- with up to 10 registered users;
- with up to three different clients' bases;
- with each database containing up to 5.000 clients' records.

Often, the user has no access to the code: most parts of the software operate as a black box and he have to call the provider to correct bugs, or simply hope for a correction in a new version. In other cases, the user has obtained the source code but has no right to modify, to change the condition of use or to redistribute it: access to the code does not make a software "open source".

Shared Source Software

The success of open source software, partly due to the market demand for more security by transparency, has convinced Microsoft to provide a better access to source code with the introduction of its specific concept: the Shared Source license (first applied to the Windows CE embedded system). According to this license, the user has the rights to access the code, to modify and enlarge it, provide it is for test purpose and without right to redistribute it commercially. Mainly on this last point, the shared source model therefore presents a fundamental difference with the OSS conditions.

Free Software or Open source Software

The concept of Free Software was born in the mid eighties, driven by Richard Stallman, founder and still today animator of the Free Software Foundation (FSF). Free does not mean "for free" as in "free beer", but is a matter of freedom and philosophy[20]: Free software exists when users have freedom to run, copy, distribute, study, change and improve the software.

The concept was used and developed in many projects, as the FSF GNU[21] project, and the Debian[22] project that established the Debian Free Software Guidelines (DFSG). It is based on a range of freedoms:

- The license cannot restrict the use (number of users, machines etc.);
- The software must be provided with its full source code (or at least this code must be permanently available and downloadable without additional costs);
- The license must allow the user to modify and extend the received code, in order to adapt the product to his needs, to improve or correct it, to enlarge it with new functions, or to re-use all or part of it in derived works;

[20] http://www.fsf.org/philosophy/free-sw.html
[21] Acronym for "Gnus' Not Unix (as a reaction against appropriation of Unix Software in mid 80s) and reference for a wide panel of free software products, popularised today around the Linux kernel developed in 1990 by Linus Torvalds. This kernel and the other GNU project components making together the GNU/Linux system, distributed by a number of IT companies.
[22] http://www.debian.org

- The license must allow the user to redistribute the above modified versions or new products (that may be under condition to mention original parts or authors, date and purpose of modifications, version number etc.);
- Non-discrimination against persons, user groups, type of use (e.g. commercial / non commercial), other software (e.g. a free software cannot impose that all other software running on the same system should also be free).

The – similar – concept of **open source software** was born later, in 1998, with the Open source Initiative (OSI)[23]. The difference with "Free Software" was originally justified by the acceptance of the vocabulary. The Free software movement image was then too much linked with "Hackers" and the concept of "Open source" was considered to be less ambiguous and more acceptable to attract the main actors of IT industry (Corel, Sun, IBM etc.) to pay attention and invest in a new business model "***From licensing to service***". This initiative was successful, as many of these IT majors now support some "open source" projects and products.

Behind the difference of terminology, there are obviously some conflicts of personalities (the strong ego of FSF and OSI leaders) and of philosophy: the OSI makes more concession to the commerce and industry, and accepts more licenses (about 30) as conform to its open source model than the FSF (about 15 licenses)

In the years 2000-2002, the "Open source Software" terminology is extensively used[24] in the professional English literature (also to eliminate the confusion about the fact that Free Software is never for free concerning the total cost of ownership). The paradox is that a lot of declared "Open source Software" is distributed with the GPL "free software" license, and that "Open source Software" is generally translated in French as "Logiciel libre" (or "Libre Software" in Spanish), which means "Free Software" in the case of reverse translation…

The solution of this "terminology quarrel" if any, seems to be in the hands of the leading organizations: the FSF, the OSI and is depending on their willingness to adopt a common vocabulary and translations in our various national languages.

A reasonable conclusion is that the importance of the "marketing terminology" is very relative: in technical documents, software should by named and classified according to their license: "GPLed software" for example, eliminates all confusion.

[23] in California – Palo Alto, with the Open source Initiative proposed by Chris Peterson

[24] We are quite often reproached to use the "open source software" terminology rather than "free software, but this wording is used in the official EU documents as the e-Europe Initiative. In the French version of our documents, we translate by "Logiciel Libre". The wording "Logiciel à code ouvert" is used in French Canada, and the wording "Libre Software" is recommended by the first European working group, but without encountering much success.

A legal approach of Open source software

As technically an "Open source" software in not much different from any other software the difference comes from the license: it is therefore the license that must be "Open source".

The key role of the license is too often forgotten: The UK government recently marked its interest for open source and launched (10 December 2001) a draft public consultation on "Open source software use within UK government". This document mentions "*Open source Software (OSS) is software whose source code is openly published, is usually available at no charge, and which is often developed by voluntary efforts*". This definition is not false but is incomplete, may be due to a "simplification purpose". It is important to note that "no charge" is not an essential characteristic of open source, and that the main key word is missing: OSS **must** be licensed and the license must be in conformity with the **Open source Initiative** criteria.

The real origin of open source software / free software is therefore the publication of the first free licenses: the BSD, or the famous GPL General Public License created by Richard Stallman[25].

The 9 conditions for open source licensing.

As open source doesn't just mean access to the source code, and as there are many different types of licenses, it is useful to remember the nine conditions[26] fixed by the Open source initiative (OSI – Bruce Perens[27]) to accept a license as "Open source". These nine conditions are the base of the OSD (Open source Definition):

1. Free Redistribution
The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programmes from several different sources. The license shall not require a royalty or other fee for such sale.
Rationale: *By constraining the license to require free redistribution, it eliminates the temptation to throw away many long-term gains in order to make a few short-term sales dollars. Without that, there would be lots of pressure for co-operators to defect.*

2. Source Code
The programme must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost– preferably, downloading via the Internet without charge. The source code must be the preferred form in which a programmer would modify the programme. Deliberately obfuscated source code is not allowed. Intermediate

[25] In a reaction against (Unix) software appropriation, Richard Stallman, formerly a programmer at the MIT AI Lab, wrote the GNU (Gnu's Not Unix) manifesto in 1983 already, where he calls for a return to the public sharing of source code. Stallman resigned from his job to start the GNU project in 1984.
[26] See also the "Study into the use of Open source softare in the public sector – Part 3 on www.eu.int.ispo.ida
[27] Bruce Perens wrote the first draft of this document as "The Debian Free Software Guidelines", and it is now a cornerstone of the OSI policy- see at http://www.opensource.org/docs/definition.html

forms such as the output of a pre-processor or translator are not allowed.
Rationale: OSI requires access to un-obfuscated source code because you can't evolve programmes without modifying them. Since OSI purpose is to make evolution easy, it requires that modification be made easy.

3. Derived Works
The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.
Rationale: The mere ability to read source isn't enough to support independent peer review and rapid evolutionary selection. For rapid evolution to happen, people need to be able to experiment with and redistribute modifications.

4. Integrity of The Author's Source Code
The license may restrict source-code from being distributed in modified form *only* if the license allows the distribution of "patch files" with the source code for the purpose of modifying the programme at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.
Rationale: Encouraging lots of improvement is a good thing, but users have a right to know who is responsible for the software they are using. Authors and maintainers have reciprocal right to know what they're being asked to support and protect their reputations.
Accordingly, an open-source license must guarantee that source be readily available, but may require that it be distributed as pristine base sources plus patches. In this way, "unofficial" changes can be made available but readily distinguished from the base source.

5. No Discrimination Against Persons or Groups
The license must not discriminate against any person or group of persons.
Rationale: In order to get the maximum benefit from the process, the maximum diversity of persons and groups should be equally eligible to contribute to open sources. Therefore OSI forbids any open-source license from locking anybody out of the process.
Some countries, including the United States, have export restrictions for certain types of software. An OSI-conformant license may warn licensees of applicable restrictions and remind them that they are obliged to obey the law; however, it may not incorporate such restrictions itself.

6. No Discrimination Against Fields of Endeavour
The license must not restrict anyone from making use of the programme in a specific field of endeavour. For example, it may not restrict the programme from being used in a business, or from being used for genetic research.
Rationale: The major intention of this clause is to prohibit license traps that prevent open source from being used commercially. OSI want commercial users to join OSS community, not feel excluded from it.

7. Distribution of License
The rights attached to the programme must apply to all to whom the programme is redistributed without the need for execution of an additional license by those parties.
Rationale: This clause is intended to forbid closing up software by indirect means such as requiring a non-disclosure agreement.

8. License Must Not Be Specific to a Product
The rights attached to the programme must not depend on the programme's being part of a particular software distribution. If the programme is extracted from that distribution and used or distributed within the terms of the programme's license, all parties to whom the programme is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.
Rationale: This clause forecloses another class of possible problem, avoiding that the license forbids or restricts rights to use other programmes or at the contrary impose to use other programmes (e.g. included in the same distribution).

9. License Must Not Contaminate Other Software
The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programmes distributed on the same medium must be open-source software.
Rationale: Distributors of open-source software have the right to make their own choices about their own software.
According OSI, the GPL license is conformant with this requirement, as GPLed libraries "contaminate" only software to which they will actively be closely linked at runtime, not software with which they are merely distributed.

Copyleft or non Copyleft

The "Copyleft" concept is a kind of "political joke" (left against right), and at first sight may be perceived as the contrary of Copyright. It is not the case: copyleft is just a specific way to apply copyright. As the FSF declares, "*To copyleft a programme, we first state that it is copyrighted; then we add distribution terms, which are a legal instrument that gives everyone the rights to use, modify, and redistribute the programme's code* ***or any programme derived from it*** *but only if the distribution terms are unchanged. Thus, the code and the freedoms become legally inseparable.*"
"Proprietary software developers use copyright to take away the users' freedom; we use copyright to guarantee their freedom. That's why we reverse the name, changing ``copyright'' into ``copyleft.''

With the "Copyleft" notion, the GNU General Public License has introduced a protection clause to avoid that GPL software may become proprietary. This clause is expressed by article 4 of the GPL license (annexe): "*You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License…*"

The copyleft protection is therefore at the origin of a kind of "***viral effect***" attached to the GPL and other "Copyleft licenses". At the contrary of some declarations, this "copyleft effect" does not touch or affect other software, or software interoperating with the GPLed software. But if you really INCLUDE significant parts of copyleft code (e.g. GPLed components) in your software, this software must be distributed under the same license (e.g. GPL license).

The Copyleft effect should be understood correctly and not "overestimated": it was once said that the greatest fear of proprietary software vendors was the inclusion (by some hacker?) of " one line of GPL code" in their proprietary software, with the effect of making the whole lot "corrupted". This is obviously excessive: the worse effort the proprietary vendor should produce is then to rewrite the contested parts using its own code. Indeed, at the difference of patents, copyright rules (protecting here the copyleft code) apply to the form of the code, not to the idea, to the business method, to the content or to the functionalities.

Not all open source software is " Copylefted". A good part of OSS is distributed with **Non-Copyleft** licenses, like the BSD license and similar (X11, Artistic…). These licenses allow the user to integrate the open source components in their new proprietary software. The component becomes proprietary within the new software environment only (its original version can still be used for any purpose by other free or proprietary developers).

The various OSS Licenses

The Open source Initiative (OSI) maintains a list of OSS conditions compliant licenses. The list is available at: http://www.opensource.org/licenses/index.html and includes more than 30 different licenses (May 2002):

- The GNU General Public License (GPL)
- The GNU Library or "Lesser" Public License (LGPL)
- The BSD license
- The MIT license
- The Artistic license
- The Mozilla Public License v. 1.0 (MPL)
- The Qt Public License (QPL)
- The IBM Public License
- The MITRE Collaborative Virtual Workspace License (CVW License)
- The Ricoh Source Code Public License
- The Python license (CNRI Python License)
- The Python Software Foundation License
- The zlib/libpng license
- The Apache Software License
- The Vovida Software License v. 1.0
- The Sun Industry Standards Source License (SISSL)
- The Intel Open source License
- The Mozilla Public License 1.1 (MPL 1.1)
- The Jabber Open source License
- The Nokia Open source License
- The Sleepycat License
- The Nethack General Public License
- The Common Public License
- The Apple Public Source License

- The X.Net License
- The Sun Public License
- The Eiffel Forum License
- The W3C License
- The Motosoto License
- The Open Group Test Suite License
- The Zope Public License
- The University of Illinois/NCSA Open source License

Five license groups

To clarify the variety of OSS licenses, the most complete European legal study about open source software[28] classify the licenses in five groups:

- Copyleft licenses (GPL / LGPL)
- BSD type
- Mozilla type
- Artistic type
- Others (specific)

Copyleft Licenses

GPL

Created by Richard Stallman, this is the most popular OSS licence, under which the software of the GNU project is distributed. A lot of software that was not initially part of the GNU project is today distributed under GPL (as the Linux kernel). The GPL was carefully designed to promote the production of more free software, and because of that the "copyleft" (art 4) explicitly forbids actions on software that could lead to the integration of GPLed software in proprietary programmes. The GPL is based on the international legislation on copyright, which ensures its enforceability. The main characteristics of the GPL are the following: it allows binary redistribution, but only if source code availability is also guaranteed; it allows source redistribution (and enforces it in case of binary distribution); it allows modification without restrictions (if the derived work is also covered by GPL); and complete integration with other software is only possible if that other software is also covered by GPL.

LGPL

The LGPL (GNU Lesser General Public License), also used in the GNU project and issued by the Free Software Foundation, allows integration with almost any kind of software, including proprietary software. The LGPL is not

[28] Till Jaeger and Axel Metzger, Open source Software, Rechliche Rahmmenbedingungen der Freien Software, Verlag C. H. Beck Munchen 2002.

a strong copyleft license, because it permits linking with non-free modules, but it is compatible with the GNU GPL.

The LGPL was originally the "Library General Public License". The FSF prefers now to recommend the simple GPL (and therefore changed the word "Library" into "Lesser", although it is better to use the Library GPL in certain cases). The most common case is when a free library's features are readily available for proprietary software through other alternative libraries. In that case, the library cannot give free software any particular advantage, so it is better to use the Library GPL for that library. Example: the GNU C library. Without the "freedom" given by the LGPL, the library should not be used anymore (as – between two equivalent solutions – the developers usually select the solution that imposes the less constraints…)

On the other hand, when the free software community constructs a library that has no correspondence in proprietary software, it recommends the GPL license.

BSD Type

BSD type includes The X-11 and other X-type license[29] as Xfree86, the BSD and modified BSD, the Apache software license[30], the Cryptix General License[31], The W3C Software Notice and License[32], The Python Copyright License[33], the Zope Public License[34], the LDAP public license[35] and the Phorum License[36]

The common characteristics of these licenses, under which some of the most used open source software is distributed - like Apache, the web server that covers more than 60% of the corresponding market - is a short text that imposes few constraints to the user. In particular, there is no "copyleft" obligation, meaning that a proprietary software developer can include the distributed OSS components in his products.

The X licenses give unlimited rights to "use, copy, modify, merge, publish, distribute and/or sell copies of the Software" sometimes adding specific conditions about copyright mention (for example, if you include Apache components, you have to mention in your documentation the wording: "*This product includes software developed by the Apache Software Foundation")* or conditions to clearly indicate in the code the date, author and purpose of modifications.

Mozilla type licenses (MPL)

With unequal rights between the original author (called "Initial developer) and contributors, the Mozilla license is also a copyleft license, but the "obligation to reuse the same MPL license" is limited to the **source code** that "must be

[29] http://www.x.org/xdownload.htm
[30] http://www.apache.org/license-1-1
[31] http://www.cryptix.org/docs/license.html
[32] http://www.w3.org/consortium/legal/copyright-software-19980720
[33] http://www.python.org/doc/copyright.html
[34] http://www.zope.org/resources/zpl
[35] http://www.openldap.org/software/release/license.html
[36] http://www.phorum.org/license.txt

published / available" to the community (and to the initial developer) after each contributors' modification.
At the contrary, the binary executable version can be re-distributed by the contributors under any license (including proprietary): this limits the legal right for end-users to duplicate / distribute executable versions, while preserving the right of developers (contributors) to develop, compile and distribute new versions.

The MPL seems therefore to offer an interesting compromise between Open source development and business. It is estimated non-compatible with the GPL, for two main reasons (in our opinion):
First the copyleft effect is different – and therefore in contradiction with the GPL: a MPL code cannot become GPL (unless by decision of the initial developer).
Second, the MPL license (section 13, Exhibit A) allow the initial developer to reserve some identified parts of the code for other (also proprietary) licensing, providing him the opportunity to release a single software package with dual licensing (for example MPL for the API and all parts related to interoperability, and a proprietary license for the core system)[37].

Artistic type licenses

The original artistic license was often reproached to be too vague, and enabling the transformation of OSS into proprietary software. However, the Perl version of the Artistic License has been accepted even by the Free Software Foundation.

Others (specific)

Other licenses are usually elaborated to give specific rights to the author. Inside this group, the Netscape Public License (NPL), the Apple Public Source License (APSL) and the Q Public License (QPL) are considered as significant examples.

We can add to this last group the CIRCA license hereafter, although it is not "open source" for specific restrictive reasons (the license is free of charge and the code is open, but the public and the redistribution (for example) are controlled.

A Public sector license example: CIRCA

Within the IDA-2 project (Interchange of data between Administrations) that runs under the responsibility of DG ENTR, Eurostat has developed CIRCA, a GroupWare Solution for public administrations.
This Internet service (library feature) allows Community services to share documents with their national networks. It facilitates the support of Working Groups. CIRCA is currently used by thousands of internal and external users linked to 22 services amongst them the EU Secretariat General.

[37] This seems the main reason why, according to the FSF, the MPL presents "complex restrictions make it incompatible with the GNU GPL, with the consequence that a module covered by the GPL and a module covered by the MPL cannot legally be linked together.", with one exception in the case of dual licensing: if the GPL is the other license…

As Member States' public administrations have shown interest in running CIRCA services themselves under their own responsibility, the CIRCA Steering Committee[38] has elaborated a **CIRCA license agreement**. This license foresees a use of the product free of charge, without any service or guarantee from the side of the Commission, and has received that approval of the EC Legal Service.

The CIRCA License text clearly identifies the Licensor (EC) and includes some frequent Open source dispositions:

- No license fee for the product
- Right to access and modify the code
- In case of modification, Licensee must respect trademark, logo, copyright etc.
- Usual limitation of Warranty and Liability: no documentation, technical support, telephone assistance, or enhancements or updates to the Product. These limitations are reinforced by specific CIRCA considerations regarding specific uses (encryption), risks (fault tolerance), "force majeure" and the possible need for purchasing other products.

Regarding a second range of points, the CIRCA license is NOT an open source license according to the OSI conditions:

- Restriction concerning the Licensee: it must be an agency or any European national administration;
- Restriction of use: for internal or external non-commercial purposes;
- No redistribution unless explicitly authorised by Licensor.
- Modifications must be made available and belongs to the Licensor
- Limited duration (3 years)

Last, the CIRCA license contains various legal dispositions that are usually ignored by " American" open source licenses:

- The governing law (Luxembourg);
- The competent court in case of litigation (also Luxembourg)
- A mention that the United Nations Convention on Contracts for the International Sale of Goods is NOT applicable
- The indication of the controlling language (English).
- Licensee is responsible for complying with any local laws
- Licensor may use Licensee's name in customer reference list or press release

[38] composed by high level representatives of DG ESTAT (chair), DG ENTR, DG ADMIN and OPOCE. For more information, please refer to Mr. Leonard Maqua – Leonard.maqua@cec.eu.int

The License agreement qualification

In general, the relationship between media distributor (that may be the author, or the POSS as intermediary service) and a user (purchaser or licensee downloading the software) is somewhat complex. On one hand there is an on-line transaction directly between the media distributor and the user, on the other hand use of the work by the licensee is subject to license terms and conditions which are set by the rights holder but are presented to the licensee by the media distributor.
One could argue that the transaction between media distributor and licensee is like a sale of goods transaction, where the "purchaser" orders a good from the seller, the seller delivers the good to the purchaser and the seller receives a price in exchange.

However there are essential differences between a sale of a good and a transaction involving digital information, licensed under the conditions of an on-line Open source license contract:

- In the first place a good is a tangible property whereas information is an intangible "virtual" property.
- Secondly the purpose of a sale of goods is to pass title in tangible property (it is un-conditional, for example you can modify your house, resell it etc.). At the contrary transaction involving an intangible property entails a license of use, where unlike a sale transaction, the transfer of digital information from one party to another is submitted to conditions.

The purpose of a transaction in digital information is not to pass title but to grant rights and privileges in the use of the information to the licensee.

If everyone agrees on the rights and obligations generated by each of the specific OSS licenses, the general contract qualification is marked by attempts to "attach" the license to other existing contracts. This qualification has no direct influence on the license content (rights and obligation) but is important as soon as a judge will interpret the license, as many case-law exist for each type of contract (sale, donation, hiring etc.). In French law, M. Clément-Fontaine sees in the GPL a variety of "Contrat de Louage » (hiring) if the license is provided for a fee, and the qualification of « prêt » (loan / loan on trust) if the license is provided free of charge (other than a possible contribution for packaging, sending or media support cost). As the idea of "loan" is attached to a limited duration (a return) we are more attracted – in the GPL case - by the concept of gift (gift with duties / donation avec charges)[39]. In all cases, the contract juridical qualification is depending on each license: "loan" is more adapted when the license is given for a limited duration (which is the case for the CIRCA license).

[39] Which is also the opinion of Jaeger/Metzger, op. cit .

The contract formation

When analysing the formation of the contract, the acceptance of the GPL (or of any other OSS licenses) conditions can be defined either as a “Shrink-wrap license”, either – in most downloading cases – as a “On-line license”, or as an “on-line contract” concluded between two parties by exchanging electronic messages relating to offer, acceptance and terms of the contract via a public network such as the Internet.

Both contracting processes have been analysed in legal literature[40].

A “shrink-wrap license” (also known as “box-top”, tear-open or “blister pack”) is based on an assumption that the user will be bound by the license from the moment he opens the shrink-wrap in which the software is packed. The problem with such licenses is that the user has no possibility to learn about license terms before breaking the seal. Although there is case law (contra and pro)[41], we will not analyse this process in depth as it should not be utilized in a POSS service.

The On-Line license is typically accepted with a mouse click prior to downloading, with three main characteristics:

- The user knows, by the way of a warning notice, that the downloading transaction is subject to a license
- The user receives (an has the possibility to read, even if he does not…) the full license terms before he continues with his downloading command;
- The user has to signal agreement by “signing” (usually by clicking on an “I Accept” button.) This agreement, if recorded, may be used as evidence of a valid contractual agreement.

Legal commentators are positive about the enforceability of on-line license[42], provided that the user is informed before ordering and that the “I accept” button should follow the text of the license (the best place is at the bottom of the licence).

[40] Trompenaars / Hugenholtz, Formation and validity of On-Line contracts – Institute for Information Law, Amsterdam, June 1998 / ISBN 90-74243-16-9 – Hereafter “IIL”
[41] IIL p. 9
[42] See:
- Kochinke / Günter, Shrinkwrap Lizenzen und Datenbank Schutz in den USA, Computer und Recht 3/1997 p. 137
- Goodger, Beta Plus for Effort 1996 – 11 EIPR p. 639;
- Raysman/Brown, Shrinkwrap licenses revisited, The New York Law Journal 13.08.1996;
- Damico/Oliver US IP law and the Internet, Section IV Online contracts and software licenses May 1996

The Identity of the contractors

One of the main questions concerning the “contract formation” with an open source licensor or with a POSS service is the difficulty to check the identity and the legal capacity and representative character of the user. This is especially the case if the license is reserved for some categories of users. “Public licensees” for example cover a wide panel of possible users, from official representatives of IT public sector departments to individuals. For example: Can an individual teacher be considered as an authorized public sector representative? How to control the possible transfer from this licensee to a new user?

In the CIRCA license, the European Commission avoids this issue with formalism (written license agreement between both parties) and with specific clauses: according to its article 1, the licensee may be (= must be) an agency or any European national administration (member states, accession counties, TACIS countries and EFTA countries).
In addition the CIRCA license forbids redistribution: “*the Licensee shall not assign or otherwise transfer by operation of law or otherwise this Agreement or any rights or obligations herein*.”

If the use of such a specific practice or license clauses is not possible (for example because the software includes GPL components and must be licensed under GPL conditions) or if the licensor wants in any case to license the software to any user without discrimination, then technical solutions may be found in requesting e.g. digital signatures, or icons to confirm legal capacity of user, etc.

The importance of certainty regarding the Judge

Most Open source license examples do not contain any competence attribution clause (*clause compromissoire*), at the contrary to most international contracts. The OSS license (as the GPL for example) is not always an international contract (for example the author, the distributor and the user of a GPLed software may be all three located in France) but it often is. If the competent judge is not indicated, this introduces a new (and useless) uncertainty factor: the judge will be assigned according to the principles of judiciary law and international private law, but in this case several judges may be competent: 1) the judge of the defender’s legal residence and 2) the judge of the place where the contract is executed.

To avoid this uncertainty, the license (or an additional agreement regarding competence attribution if a standard text as the GPL is used) should determine the competent judge.

This is done in article 13 of the CIRCA license as follows:
“Unless otherwise agreed in writing, all disputes relating to this Agreement (including any dispute relating to intellectual property rights) shall be subject to final and binding judgement rendered by the competent courts of the Grand-Duchy of Luxembourg”.

The importance of certainty regarding the law

Most Open source license examples are American text and do not contain any law attribution clause, as the law in force is – obviously – the law of the United States.
In the case of a license governing the pooling of software elaborated by (and mainly or exclusively for) the European public sector (and no matter if this audience is finally restricted or not), it is important that the law is European too. As European directives in the matter are translated in national law and against a background of national "acquis" concerning the contract law in general, a specific Member State law must be selected. Both parties' rights and obligations gain in certainty if the selected law and the competent judge belong to the same judicial order.

This is done in article 13 of the CIRCA license as follows:
GOVERNING LAW. This Agreement shall be governed by the Law of the Grand-Duchy of Luxembourg. … This Agreement shall not be governed by the United Nations Convention on Contracts for the International Sale of Goods.

The importance of certainty regarding the language

Most of the OSS licenses are in English only. In some cases, as for the GPL, many translations exist, but the publisher (at the request of the Free Software Foundation) mentions that the translated versions are given for information only and do not have any legal value (the aim of these restrictions is to minimise the danger to have wrong interpretation or to create an unexpected situation if a "non – FSF" text like "the GPL was presented in court").

Three types of situation may occur:

- Only one version is communicated / exists between the licensor and the user (it is usually in English). This solution prevails in most OSS downloading
- Two or more linguistic versions exist, but one of these is declared as the " official one".
  This solution is the one adopted by the CIRCA license, where two versions exist: German and English.
  Art. 14 of the CIRCA license said: "*The controlling language of this Agreement is English. If Licensee has received a translation into another language, it has been provided for Licensee's convenience only*."
- Several official versions exist (in various languages). The versions have been translated by authorized and competent translator offices (jurist linguists, or in French "traducteur jurés") and all versions (the version written in the user language, as it is the case for the European legislation) may be used in court if needed. This solution was adopted by some established international software vendors.

The third solution, according to the European consumer protection, is obviously the best. It may also be the only "legal" one according to some legislations: Since 1975[43], the French law makes the use of the French language mandatory in all commercial transactions including the import of goods and services. M. Clément-Fontaine, in her study about the GPL in French law[44], indicates – Nr 40 – that the GPL is an international contract with trans-border effect, as the contracting parties are « generally belonging to several nationalities ». On this specific point, we do not follow the author, at least in the hypothesis of an European POSS : Already today, the GPL and other « American – English written » contracts are used inside Member States national borders, often between contracting parties of the same nationality (e. g. France) and this present a major issue in terms of validity according to national law.

Therefore, a European POSS should make the effort to provide « Official » versions of the approved licenses in the various official European languages. As the Copyleft concept imposes the use of the GPL as soon as GPLed code is used to distribute software, this point should be discussed with the FSF.

The Dual Licensing

Each original software author willing to authorize other users to benefit from his work can freely determine the type of license. By original author, we mean the author that has exclusive rights to the programme (that is not the case if this author received the code from a previous author, under the terms or a licenses imposing obligations, as the GPL).
As Licensor, this original author is not obliged to give equal rights to all users and can therefore use several (two or more) licenses. This is "Dual licensing". Dual Licensing has been used to separate "commercial" use for selling commercial services or for reselling the software, from private or non-commercial use.

The Copyleft effect attached to the GPL does not restraint original authors freedom to distribute under several licenses (e.g. GPL and proprietary). Can the author return to proprietary after distributing under GPL? Yes: the original author can restart from his original development, especially if – due to the complexity of the programme – the original licensor has kept a strong control and de facto exclusivity on the code[45]. However, the original author (or copyright owner) will have no rights to profit exclusively from other authors' GPL improvements "as is", and will not be able to contest the right of an open source community to continue the development and distribution of the GPLed version (possibly under another product name).

[43] Revised by the law of 4 August 1994 and decret n°. 95-240 of 3 March 1995.
[44] The Study of Mélanie Clément Fontaine is available at : http://crao.net/gpl/gpl-TITRE.html etc.
[45] For example, this is the case for the StarOffice office suite. It was purchased by Sun in 1998, was released for free under GPL in October 2000 (version 5.2) and was then downloaded by millions of users (but mainly for test purpose, as few organisations have really switched to it, and with very few users that were able and interested to enter in the complex code). Having gained this popularity attached to its StarOffice Trade Mark inside the Open source community, Sun plans to release the next version 6.0 under a specific proprietary license model, but a new OSS version "OpenOffice" was released in May 2002.

Legal background of Copyright

The Copyright (in Common Law, UK, US) or for the continental European countries the « Droit d'Auteur » (France, Belgium) / the "Urheberrecht" (Germany / Austria) etc… automatically and implicitly protects all intellectual creation, including computer software.
Copyright protection is organised by an international framework of conventions, but each country has its own implementation of these conventions.
The copyright holder does not hold a "uniform world wide copyright, but holds a bundle of national copyrights[46]. The effect of copyrights depend of the law of the country providing protection (Germany, France, United Kingdom etc.), and Anglo-American or continental laws will have different attitudes regarding key aspects, such as the distribution of the copyright revenues: to the physical author in French law (inaliénabilité du droit d'auteur) and to the "Copyright Owner" (an American notion meaning the original author or any new purchaser) in US law.

The first international convention – creating the automatic protection - was signed in **Bern (Switzerland) in 1886**. As it was not ratified by the US and the USSR, another convention (Geneva – 1952) was temporarily preferred (because it was also signed by the US), imposing some "formalism and bureaucracy": the famous © mention, indicating the work was formally "deposited" in the competent office of its country of creation, but after US ratification in 1988, the Convention of Bern became again the European Union reference). It is now being revised by an international diplomatic conference (Geneva 1996…) making of Copyright a real international and common subject of law.

The term of ***creation*** means that the protected work must be "original"(reflecting the specific author contribution and personality, even if the work is elaborated with previous material).
The term ***protection*** means that – without author authorisation – all reproductions or generally all usages are prohibited (this is the reason why a license MUST exists).

The absence of formalism is not a guarantee against litigation. Although contestations are really rare, a deposit in an author society or any publication ensuring a certain time stamp is recommended.

[46] Axel Metzger and Till Jeager, Open source Software and German Copyright Law, CH Beck Munich 2001, p. 58

What Copyright law is applicable in Europe?

Which Member State national law should be applicable? If not specified in the contract, a Member State's national law will be applied if the contract has a close connection with its territory. This is the case if the user has his business activity and his residence in the Member State, and if there is evidence that the contract was concluded in this state. For example, if the open source software is downloaded in Germany from the Internet, by a person pursuing his business activity in Germany, this is evidence that the corresponding contractual manifestation of intent is issued in Germany. Therefore, in our example, the German law is applicable.
The applicability of a European law to evaluate the dispositions of an American text will be subject to interpretation by the European judge: to stay with the GPL example, it includes in its section 1 the American notion of "Copyright Holder" where the European laws consider only the author (inalienability of author's rights).
The issues attached to the identification of the law are clearly reduced if a valid contract determines the governing law.

What is protected?

In copyright, the form is protected. In contrast to a patent, the idea, the business methods, the content or the functionalities of software are not protected by copyright. This is the reason why, when protecting copyright, the open source license does not impose excessive constraints on proprietary software industry. As was said in the "copyleft" comments, the viral effect of a license like the GPL is a myth in the sense that it will not constraint a publisher to release his software as GPL if some lines of open source code where introduced in it by accident.

Who benefits of copyright

The beneficiary of this protection is the author: in the case of software this author may be the programmer (or a group of programmer) but – if the programme was developed in the framework of a contractual mission – it may also be the employer (including public authorities) or the customer (also including public authorities) according to the specified conditions. There are no general rules as the determination of the author may vary according to circumstances and to the contractual conditions, but in all situations there is an author, and a public sector authority may be « The author ».

The consequence of copyright is that the distribution and use of software (including the Internet distribution) must be authorised by an explicit « action ». The expression of this action is a **License Agreement**, that is a standard contract proposed to the users (no matter if the contract is given for free or for a fee) indicating the rights and possible obligations that are imposed to these users or "purchasers".

Conformity to Droit d'Auteur in France

According to the variety of open source licenses, each of them should be examined separately. However, the most critical license is the GPL since its "Copyleft" concept is a "newcomer" in the organised world of copyright, giving both wide rights to modify and redistribute and precise obligation to do it according to the same GPL conditions. The conformity of the GPL to the « Droit d'Auteur » continental Europe concept was admitted by the study of Melanie Clément-Fontaine[47]:
The GNU GPL conditions are consistent and reliable regarding the French contractual and copyright laws, and moreover it is in conformity with the « spirit / finality » of these laws : « *la licence publique générale GNU organise un mode d'exploitation de logiciel fiable au regard du droit des obligations et du droit d'auteur, et est conforme à sa finalité* ».

The novelty of the GPL regarding copyright (and more generally of most OSS licenses) is that they introduce an unprecedented notion of « **movement** » and a new logic applied to well established law principles:

- The software (object of the license) may be modified
- The author community (contributors to the progressive software evolution) may be modified (extended)
- There is no specific time/space constraints (or duration)
- The license text itself (transmitted by each successive author to its followers) is also subject to versioning.

The « **logic** » of copyleft is also new, since the « ***exclusive property right*** » foreseen by copyright law for the benefit of the author is reversed to grant a « ***general non-proprietary right*** » to a community where everybody can add, but nobody can withdraw. But doing that, the GPL is in conformity with contractual law (la *licéité et la prévisibilité des conventions*) as with the letter and the spirit of the CPI (the French "*Code de la Propriété Intellectuelle*"): Indeed, the GPL copyright owner controls:

- the principle of making the software publicly known (article L121-2 CPI);
- his "paternity" on the software (article L121-1 CPI);
- the respect of the rights (article L121-7 CPI)
- the reproduction and representation (article L122-6 CPI)

[47] La Licence Publique Générale GNU, DEA by Mélanie Clément-Fontaine, Université Montpellier I (laboratoire CNRS ERCIM), 1999. on http://crao.net/gpl/gpl-Contents.html

The validity of OSS licences and the GPL in particular according to the "*Droit d'Auteur*" concept was not obvious at first sight, as the « logic and objectives » of copyleft are totally ignored by continental law, where all philosophy (and in France in particular) is based on the "untouchable" character (inaliénabilité[48]) of copyright: the fundamental bias of OSS seems there to be in contradiction with the continental "*Droit d'Auteur*". In the new GPL logic, the "software" prevails on the "author" but this is entirely due to the original author decision (and after him, the decision of all other authors choosing to increment the same GPL source): The decision to distribute with an OSS license belongs therefore to the "moral prerogatives" that are given to the author by the Bern convention. The initial author is free to make this GPL choice, which is formally expressed by the license.

Conformity to Urheberrecht in Germany

In their book « Open source Software – Rechtliche Rahmenbedingungen der Freien Software », Axel Metzger and Till Jaeger (the German ifrOSS promoters) come to the conclusion that the GPL is globally in conformity to the German copyright law[49]. The authors' position is also expressed in the English document "Open source Software and German Copyright law"[50]
The conclusion is that most OSS licenses and the GNU GPL in particular may be applicable without major conflict with the German Copyright Law although some terminology used is questionable.

Our conclusion is that the GPL respects the Copyright principles. It is a copyright license. It is not a waiver of authors' rights that would then fall under public domain (and this should be questionable regarding the different European copyright laws). On the contrary, copyrights are maintained, by the fact that simple rights of using the software are ***given*** to everyone under ***precise conditions*** (that all modifications must also be made available under the open source software license).

The GPL is compatible with the need to safeguard the integrity of a piece of work and with the right to claim authorship. Indeed, even if modifications are authorised (as programmes are serving primarily utility and functional purpose rather than pure artistic work), the section 2a of the GPL obliges the licensee who has changed the code to highlight modified parts with an explanatory notice.

Conformity to patent law

[48] The untouchable character of "Droit d'Auteur" is expressed by article L.121.1 of the French CPI:
« L'auteur jouit du droit au respect de son nom, de sa qualité et de son œuvre.
Ce droit est attaché à sa personne.
Il est perpétuel, inaliénable et imprescriptible. »
[49] Mainly §§ 69 a) and follower of the copyright law (Urheberrechtsgesetz) as introduced in 1993
[50] Metzger / Jaeger – Open source Software – Verlag C.H. Beck oHG 2001, D 81801 Munich Germany

Open source licenses (The GPL and other) are not governed by patent law (they are governed by copyright as we have seen), but they are concerned by it.
The growing extension of software patentability, in US first, then on the European practice, is becoming a major preoccupation for OSS providers:

The reason is that patents may protect not only the form, but the use of algorithms, techniques and methods that are difficult to ignore or bypass. Patents create few or no problems if the protected software is closely incorporated, or embedded in a material device, making the whole « invented material product" protected. On the other hand, if the software industry can patent pure business methods that can be used in any programme for any purpose and by any programmer (for example, to ensure interoperability: import and export from their OSS programmes to proprietary standards), damages to the programming activity (including open source) may be important, and patent law will be distracted from its original protection purpose to become a weapon in major industry actors hands:

- Independent programmers and SMEs can hardly compete with major players in the field, and have no resource to make « cross patenting » ;
- By nature (availability of source code) OSS is more exposed to lawsuits.

Patents on protocols required for interoperability is a blocking factor when trying to implement compatible non-proprietary solutions.

Examples:

Samba (which is a GPLed software) provides file sharing interoperability allowing Unix-like machines to act as server in a Microsoft environment, using for that the Microsoft Common Internet File System (CIFS) protocol (also known as a subset of the SMB Server Message Blocks protocol).
In March 2002, Microsoft introduced a new license type, combining licensing of the CIFS technical standard and patent claims (as two patents where taken concerning the CIFS): the "Royalty-Free CIFS Technical Reference License Agreement",[51] allowing third parties for free use of the CIFS protocol at the condition that they do not use the GNU GPL and similar copylefted licenses considered by Microsoft as "IPR Impairing License". Samba answered[52] that it was unaffected by the existence of these patents, because these covered an obsolete section of the CIFS/SMB protocol. This answer, if verified, leaves the patent problem intact for further interoperability issues, as – to ensure interoperability – the dominant system protocols need to be emulated in compatible solutions.

[51] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnkerb/html/Finalcifs_LicenseAgrmnt_032802.asp
[52] http://us1.samba.org/samba/ms_license.html

Another recent example of disturbing patent claim[53] is related to ebXML: The standard was launched in 1999 by a United Nations group (OASIS) in order to promote Internet based trading, and in may 2000 IBM provided a substantial contribution (TPAML XML formats to standardize trading partnership and electronic contracts). This was incorporated into ebXML V 1.0 (May 2001) and adopted by partners (RosettaNet, Open Travel Alliance etc.). Two years after (March 2002) IBM suddenly informed OASIS that its contribution was patented. Here also the fact IBM finally decided to offer the TPAML at “zero license cost” leaves the potential legal issue intact and demonstrates the difficulty to investigate on previous patents – even concerning a world wide promoted standard.
These considerations (very briefly summarized here) are the reasons of the strong opposition of Open source communities towards any extension of software patentability, as it is foreseen in the directive proposal of the European Commission (February 20, 2002).

A practical consequence of software patentability regarding the publication or the pooling of open source software inside the POSS is the requirement to investigate on possible patents, in order to avoid legal hassles and even higher costs. This issue will be considered by anyone who publishes software (free or not), but – due to the absence of cross patenting and of important license incomes - will be less favourable to “small and medium enterprises” and to free software developers.

[53] See Computer World 22 April 2002 p. 1 “*IBM claim to ebXML patent*”

The Contractual framework

The POSS contractual framework may be organised on the following structure:

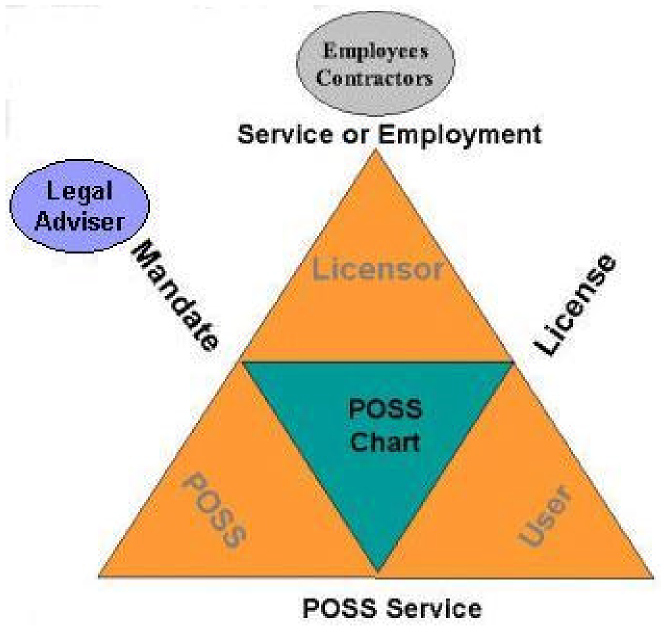


Figure 7 Roles and contracts

Poss Chart (or "general terms")

The Poss Chart is the POSS service "statute", publicly available on the POSS site, describing the aim and object of the POSS, its organisation and who is behind it, its general rules of operation, its ethics.
As mentioned in the "Maintenance chapter" the role and authority of the POSS Scientific Committee and of the site administrator must be described.
The Chart will stress on European Union cooperation spirit in sharing software within the respect of cultural diversity and "subsidiarity".
It may mention, for example:

- The neutrality of the POSS (regarding commercial competition and philosophic debates between persons and organisations)
- Role definitions (Licensor/donator, Licensee/user, POSS)
- That licensors, their product and documentation referred by the POSS service, the users (in their participation to POSS forums) should avoid writing or saying anything which adversely affects the good name of other public or private persons, products or companies;
- That the Scientific Committee will have all authority to correct or remove any information that seems not fully compliant to the POSS spirit, without having to motivate its decision.
- The engagement to respect citizens, governments and enterprise privacy
- The respect of fundamental European legislations (Treaties, Human Rights Convention etc.)

Mandate

The contract of commission (or mandate, or “donor contract”) will organise the relation between the Licensor and the POSS.
The mandate is especially important if the Licensor wants that the POSS should represent him in contracting with users (otherwise – if the Licensor wants to authorise users himself from case to case) the POSS will limit his mission to information and linking.

Points of this contract are:

- Identification of the software
- Guarantee given by the licensor that he is the copyright owner
  - That the software was purchased or developed according the terms of a **service contract** providing distribution rights to the licensor, or:
  - That the software was developed by licensor employees under **contractual terms giving the copyrights to the licensor**.
- Commission given by the Licensor to the POSS to distribute the software.
- Commission given by the Licensor to be represented by the POSS when contracting the license with the user
- Indication of the selected license (see hereafter) with declaration regarding the license (e.g. if Licensor chose a “Non-GPL” license, that its software was not constructed from GPLed components)
- Indication of license price (or for free)
- Indication of possible other specific conditions
- Indication of optional services and conditions (documentation, support)
- Guarantee exoneration clause regarding
  - Direct/indirect damages to user
  - Copyright, Patent or other third party rights violations
- Acceptance of POSS conditions regarding applicable law and jurisdiction.
- Engagement of the author to maintain and update the project information (e.g. every six months)

Acceptance by of the author of the authority of the POSS Scientific Committee regarding all information published on POSS pages.

Legal adviser

The POSS itself will not issue legal advices, but a service level agreement with an external specialised legal adviser may be a useful service to provide to candidate licensors, especially in copyright and patent inquiries. The professional liability of this legal adviser will be granted by an insurance company.

An single external “open source” specialised legal adviser presents more guarantees of consistency in advices than the various donators legal departments (that are less specialised in open source and may try to cover their liability with excessive protections in specific licenses).

License

The license is the contract between the User and the Licensor.
As the POSS infrastructure will contain the tools (web site, certified process of contracting procedure), it may simplify the Licensor task to be represented by the POSS when contracting with the user.
Prior to contracting via the POSS, the licensor will have to indicate one (or more in the case of dual licensing) license and conditions.
The selection of a license is covered below.

POSS service

The POSS service contract organises and clarifies the relations between the user and the POSS / The POSS as representing the licensor.
From the user point of view, the POSS service contract may be additional to the license.
Another solution is to "embed" the license into the POSS service contract, at the condition it will not cause confusion between the POSS and the Licensor.
In all cases, the POSS service contract should never contradict any point of the license (in particular, copyright and other points regarding intellectual or industrial property should not be present), but will add precisions where the license agreement is dumb.

Points of this contract are:

- Indicate the respective roles of the POSS and the Licensor.
- Refer to the POSS chart (aim and spirit, role of site administrator and Scientific Committee).
- Indicate that the service is provided as a gift (free service).
- Exclusion of POSS liability for direct or indirect damages.
- Acceptance of POSS conditions regarding applicable law and jurisdiction, concerning both the POSS service and the License (creating a uniqueness of jurisdiction for all possible litigations).

The choice of a license process

The license Type

The existence of so many licenses may leave the authors with uncertainty. In reality, many licenses are variants of a very few number of “core licenses types”. 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[54]

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 include your source code into 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 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.

[54] This grid is a modified version from the quick reference for choosing a free software license, version 1.0.4, 2002-02-08 http://www.zooko.com/license_quick_ref.html; for an interactive version, see: http://yoyo.org/~pgl/lqr/

"2" MPL 1.1 can be specifically amended to allow combining with GPL, according to the FSF's license list.

Comments

- Question 1- evaluates whether members of the developers community like to *use* (your) source code under this license, instead of whether members of the community like to *create their* new source code under this license. This assumes that the public sector author has already created his own source code (e.g. from scratch, if you compare with question 6), and that he wants his source code to be used as widely possible by other public sector developers (or more widely by members of the open source/free software community). The difference between these two meanings of "likes it" is shown up by the case of the GPL: a hypothetical re-developer, wanting to make and commercialise derived works, may prefer to *re-use* source code licensed to him under a license that permits her to combine the licensed source code with proprietary source code. This is the reason why – although all the license types respond to the question – the GPL will be less attractive than the BSD for software industry, because less commercial freedom will be given to them (this may be your objective too).
- Question 2 "Do you allow Vendor enterprises to include your source code into their (proprietary) products and to redistribute" investigate if it should be accepted that any person or more precisely a software vendor should combine your code with his own proprietary and redistribute it as proprietary code (even if he has no exclusive rights to your public sector code: any other enterprise could do the same).
  If you answer "Yes", you will not distribute under the conditions of the GPL (that prohibits proprietary redistribution), but you can choose a BSD, LGPL or MPL type license
- Question 3 "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. If you answer "Yes" this mean that you do not want to preserve the original combine with GPL'ed code and redistribute" -- It is legal to accept code from its author under the terms of this license, combine it with GPL'ed code, and ship the resulting application to a third person.
- Question 4 "Are other programmers modifying and redistributing your source code obliged to publish / share the source of their modified or redistributed version ?" Does your license forbid the recipient of the source code from modifying it and shipping his modified version to a third party without giving them the source? If you want to forbid that, do not choose the BSD

- Question 5 "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 ?" - Does this license require that if the recipient combines the code with his own contribution and then ships the resulting combined application, that he must contribute a license to any patents that he holds that would restrict usage of the resulting application ? This is a logic consequence of the two "Copyleft" licenses. Indeed, if you provide the code, what is the use for third parties if this code is patented ? Copyleft has been created to avoid this "proprietary despite Open source" situation. If you answer "Yes" to the question, a copyleft license (The GPL) is recommended.
- Question 6 "is your code based or including one or more pieces of code that was obtained under GPL License? If the answer is "Yes", then you have no other choice than the GPL (consequence of "Copyleft").

A the LGPL is definitely less usable[55], a simplified choice may be limited to 3 types of licenses: the most permissive BSD and two copyleft licenses: GPL and MPL

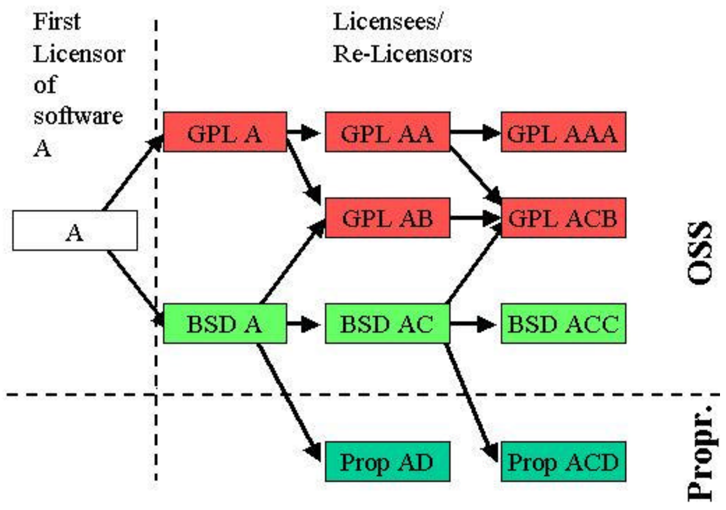


Figure 8 GPL / BSD impact

The difference regarding re-distribution 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 BSD, he put the code at the disposal of the community with almost total freedom: the same code may

- become proprietary in its version AD;
- 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.

[55] The FSF itself recommends to use the GPL instead

In other words: the GPL is a gift to all users and to the "**free software developers community**", the BSD is a gift to all users and to "**all developers community**" without any restrictions (and therefore may be more liked by the software industry)

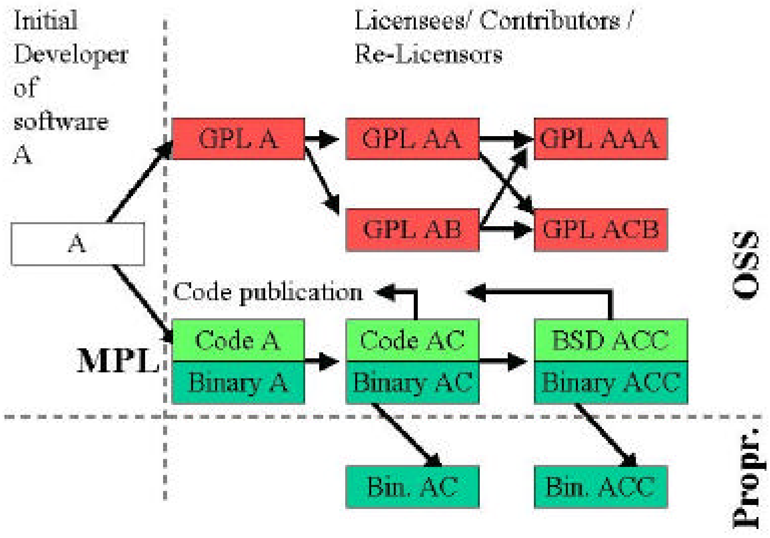


Figure 9 GPL / MPL 1.1 impact

If an MPL license is selected, the situation is very different: apart from the initial developer (who can change his mind later and release – e.g. – under GPL), all later contributors must stay with the MPL and publish their modifications (informing the initial developer) in the case of sub-licensing. There is no bridge to GPL anymore regarding the code, and binary versions can be delivered as proprietary package with no redistribution right (provide it is indicated that the source code is available at address "x" under MPL license).

The MPL 1.1 appendix also allows the licensor to "reserve" the property of identified parts of the code, which may be necessary if the developed solution was partly based on proprietary components.
The MPL 1.1 is considered as an interesting compromise: the code is open for developers and contributors, but binary redistribution is controlled.

European Public Sector License?

A supplementary question (Question 7) may be issued "Do you want to reserve the use of your software to (European) public sector?"
If the answer is "Yes", there is no other solution than to elaborate a new specific European Public Sector License (EPSL).

This license could be inspired by the CIRCA model (provided in appendix), removing the specific mentions concerning CIRCA and reducing excessive restrictions concerning the duration (3 years with CIRCA) and redistribution at the same conditions.

Such a license may be qualified as “open” concerning the access to the code and, if given for free, should fill at least one of the POSS objective: sharing software in a “best value for money”. However, it will not be considered as an “OSS license” according to the OSI, due to restrictions regarding the user group, and is not compatible either with the use of previous copylefted code (GPL or MPL).
In addition, it imposes a strict control of distribution and redistribution (identification of all successive beneficiaries, that may only be obtained by electronic signature authentication or – why not – by signing a classic paper agreement).
Therefore, we estimate that the management of such license is too complex. It may be replaced by a POSS membership identification, limiting the first access to identified public administrations only (without controlling then the subsequent re-distribution if any).

Dual agreement

If there is no other solution than to start from an American text (This is mainly the case if the GPL or MPL are chosen) it will be necessary to complete the licensee agreement with a specific agreement concerning at least the indication of the competent judge and of the governing law.
Such agreement should not alter the GPL/MPL text, as these matters are not covered by the license.
Another issue, as we have seen, is to extend this specific agreement to the language of the agreement, but that may have an influence on the license text itself, and therefore may requires an agreement of the “license copyright owner” (at least concerning the GPL text that is copyrighted by the FSF).

Liability issues

Author liability

What if a public sector agency distributes – for free – software out of its own production?
Software can produce two categories of damages:

- Direct damages (if information is erased or damaged, its replacement, restoration, intervention of technician, time directly related to the error correction and to the reconstruction of lost data, etc.);
- Indirect damage (loss of activities, loss of image, poor performances due to IT trouble and loss of confidence towards the IT system, cost of time not directly related to the error, but related to finding an alternative solution, the obligation to migrate again all data to the new solution, the training of the personal to another product, etc.).

The open source character (and transparency) of the software is not a guarantee against damages, since most users do not open the source and cannot understand all implications of native code. In addition, the most "transparent" programme may operate perfectly until unforeseen environment, parameter, conditions, involuntary or even intentional misuse could be harmful to persons, properties or information.

Without a license agreement attached, a government agency may (theoretically) be made fully liable for any consequences (direct of indirect) of bugs and malfunctions of the delivered programme.

The first indication is therefore to avoid distribution of software without a clear license that excludes the liability for these direct or indirect damages. As all open source software licenses contain such disposition, GIGA estimates that there is neither special liability nor protection from liability in using an Open source License[56]. For example, the GPL contains this paragraph:

BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

[56] IdeaByte . No Special Liability nor Protection From Liability in Using an Open source License RIB-112000-00029 - 2000 - Giga Information Group

Similar exclusion clauses are found in other OSS licenses, and on this specific point, it is significant to remark that proprietary licenses (as for example, the Microsoft EULA license) include very similar or identical wording. The inclusion of such clauses appears then as an "automatic reflex" and, although mainly due to the consumer protection law the validity of these clauses is questionable, OSS licenses do not present significant differences with others.

Validity of exclusion clauses

The global exclusion of any liability found in OSS licenses (but also in proprietary licenses; this is not specific to OSS) is not valid according to the consumer protection and the contractual law.
The real liability must be appreciated according to several scenarios, in particular if the software is not given for free.
If the software license is not charged to the end user (even if additional services, support of the delivery of a media cd-rom / tape etc. is charged) it may be considered as "a gift".
Therefore the exclusion of liability is valid provided that there is no intentional or severe negligence
We consider that, despite all liability exclusion, the author's liability is engaged if the software is damaging by nature: a classical example is the introduction of a computer virus (or an hidden backdoor) in the code.

Jaeger and Metger[57] isolate 6 different scenarios that may generate various approaches for a possible liability:

	Case	**Type of relation**
1	The software is downloaded directly from the author (author's site)	Bilateral Author - licensee
2	The software is obtained from the author on any IT support: tape, floppy, CD etc	Bilateral Author - licensee
3	The software is downloaded from a third party server (e.g. a distributor)	Trilateral Author – Distributor Distributor – User Author - User
4	The software is obtained from a distributor on any IT support: tape, floppy, CD etc	Trilateral Author – Distributor Distributor – User Author - User
5	The software is elaborated for specific internal or external purpose (or inside a contract between an Authority and a software developer to provide an OSS to its citizens)	Trilateral Authority – Developer Authority – End-user Developer – End-user
6	The software in embedded in hardware	Bilateral Vendor – Buyer

[57] Op. cit pp. 137-174

The POSS scenario is close to Nr 3 (and maybe 4 above) – if the POSS directly provides software to download from its FTP site (case 3) or if the POSS distribute software (from CDs for example)

It may also be scenario 1 (or 2) if the POSS just establish the relation between the interested user and the author, when the software is downloaded from the Author site, or is obtained from the author on a media.

The intermediary service liability

An issue related to the potential liability is the identification of the "responsible vendor", and here the OSS model presents particularities: in the case of free downloading of cooperative open source software, any kind of evidence will be difficult to provide. The source code itself – if downloaded – can provide more evidence, with its copyright indications and annotations.

The author or distributor quality (public sector agency or commercial vendor, identified and financially solvent) generally reinforces the liability: IT professionals are supposed to act with competence and experience, according to the state of the art in their branch. Private end-users or even non-IT commercial enterprises will benefit (in Court) from a reinforced protection.

Such case of liability for Open source distribution has never been tested in court (and thus, some will argue that the "legal scarecrow" is perhaps more a paper dragon than a real one). However, the core function of most public sector agencies is not to distribute software, except if it is closely related to their role (E. g. an income declaration checking software, distributed by the competent tax administration).

Therefore, if an administration wants to license software (e.g. according to one of the OSS licensing models), it should better not license it directly to end-users, but – on a non exclusive base – to "First level licensees" (a POSS identified member being a dedicated competent public agency that will check the inoffensive character of the software or/and a panel of specialised OSS vendors, association or user's groups which will organise distribution and downloading).

Beneficiaries and competition with software industry

The risk of unfair competition with software industry may exist if the public sector intervention creates serious and un-necessary damages to the software market.

Open source software in general are now well integrated in IT industry policies: many service and software providers have developed their own OSS licenses to distribute products (Sun, IBM, Apple etc.) and even the pure "License fees vendors" develop for OSS platforms (e.g. Oracle) or release specific OSS solutions (e.g. SAP with their database).

Regarding the POSS, the risk may be limited by several considerations:

- The subject (public sector specific software with less direct or massive market impact);
- The beneficiaries (public sector members only if a restricted downloading access is implemented);
- The partnership: The POSS creates new partnership opportunities with the private sector (constructing new developments on established best practices) and will liberate public sector budgets for more quality and integration;
- A comprehensive licensing policy:
  Where the BSD license is selected, the POSS will provide proprietary software development opportunities to the software industry (without impeaching other developers to start from the same code to deliver open versions).

One of the conditions of the “non-competition” is the POSS neutrality regarding platforms and possible use of proprietary components (e.g. some public sector solutions require the use of an Oracle or MS/Access database).

Concerning the beneficiaries of the POSS, it should be limited to registered users (e.g. developing for public sector in a flexible sense). In a first stage (observation period) user registration may be limited to European public sector. In a second stage, the POSS can bring a substantial contribution to emerging countries IT development (e.g. in Africa, Asia) depending on European authorities decision.

Limitations should not concern the end user itself and therefore must not be included in the license: it is difficult for legal reasons (we will see that, due to the “copyleft” effect, original OSS licenses as the GPL and MPL must be used without modifications in some circumstances) and also for practical reasons: in an open source environment allowing re-distribution, the end user control is difficult.

The possible sub-sequent software redistribution by initial licensees is depending on the license and will not be systematically registered or controlled. It will stay outside the POSS scope and liability if any, as other operation done on sites that are just “referred” by the POSS by URL links.

Some POSS service (news, forum, etc.) should be accessed by anybody, but member authentication should be required for specific services as downloading and to grant a reliable contractual trail.

Conclusions regarding the legal framework

The contractual framework is not limited to the software license, but includes:

- The general terms of the pooling service (what we call the POSS chart);
- The contract between the author and the POSS (what we call the mandate or “commission”, as the POSS will represent the author – licensor when contracting with the user - licensee);
- Specific agreements related to liability, competent judge and applicable law, patent issues;

- The license agreement itself that should be selected by the author – licensor (and accepted by the user – licensee).
- Service level agreements with various POSS service actors, and among them, one or more legal advisers to help candidate licensors to solve licensing (copyright) and patent problems, as each case is or may be specific (no general – all purpose – answer exists)

The contractual trail process with all actors must be carefully registered (user authentication, time stamp, contract archiving) in order to produce contractual evidences if needed.

Concerning the choice of the licenses, this is and will stay the **responsibility of the licensor**. The POSS may help or propose simplified choices between 3 licenses:

- BSD to give the software to all users and to "**all developers community**" without any restrictions or discrimination regarding the use (from the same original contribution some licensee may develop proprietary software, and others may continue with Open source);
- GPL to give the software to all users and to the "**free software developers community**" only;
- MPL variant if for any reasons some components must stay proprietary or if the initial licensor wants to benefit from a close follow-up of all code modifications, and reduce "piracy" (meaning there the simple duplication of binary package, without contribution or added value).

The licensor must be free to select or to elaborate another license, as long it is compatible with the POSS chart (general terms).

3. POSS Functional Requirements

Objectives of the chapter

Beside the processes and resources (chapter 1) and legal aspects (chapter 2), a third important and essential aspect is to measure the level of interest of administrations for the various "Knowledge Domains" to share in the POSS service.
Several points must then be considered:

- The way to present the POSS;
- The sharing organisation in user groups;
- The way each administration can itself contribute to the POSS.

The objective of this chapter is to identify the particular requirements of the POSS organisation in order to manage the development, the deployment and the effective support of the OSS Solutions for the administrations of the European Community and of its Member States.

Starting from the objectives of the POSS Portal site, we can define the functionalities of interest for the future POSS users. Besides offering functionalities to its users, the site must also provide system management tools for its own setup, operations and maintenance.

Objectives of the POSS Portal Site

When setting up the POSS Portal site, we must keep in mind its main objectives which are:

1) To ***provide useful solutions*** to the IT departments of the administrations of the European Commission and its Member States ;
2) To ***create a community*** of collaborative developers, users and policy makers for administrations by allowing each visitor to identify :
    - His role (licensor, expert, service provider, user…)
    - His legal framework (see chapter 2)
3) To facilitate the realisation of e-Europe 2005 objectives and among them the implementation of an European Framework of **interoperability standards** (trans-border knowledge exchange to avoid that 15 member states develop non compatible standards)

4) To ***promote the use of OSS*** and increase the size and the activity of the community;
5) To identify and ***disseminate information*** about interesting projects and software already registered on similar sites;
6) To provide a ***virtual meeting place*** for potential and effective OSS users of the European Commission and the Member States.

Providing Useful Solutions

User Requirements

In order to best meet the needs of the IT departments of the administrations, it is important to identify the needs of those administrations and the areas where a common IT solution could be used. Therefore, forums and surveys should be organized at least on a yearly basis.

In the frame of this study, we developed a questionnaire that has been distributed to several IT responsible in European administrations. The addressees were selected among the participants of the "OSS Symposium" held in Brussels on 22nd February 2001 and the developers of applications that received e-Government label at the e-government conference held on 29th-30th November 2001.

A copy of the questionnaire and some of the responses received can be found in Annex A.

Though all these administrations showed an interest for the present study and accepted the principle of sharing existing software, the most precious input could only be gathered from the countries that already had a good level of expertise in the pooling of developed software and applications, such as France, Sweden, Denmark and Germany.

Here is a summary of the input from those administrations:

- Many projects developed could be used in other administrations (data exchange, groupware, human resource management system with a web interface) but today there is a lack of exchange means.
  A European POSS portal should take into account sites providing the same services that have already been developed at the national level, for example by providing links to those sites.
- A quick reference guide for the selection of the license model should be available on the site.
  Standards (coding, data format…) should be defined on the POSS portal and all the projects registered on the site must be compliant with them.
  Security aspects have to be carefully examined : both software security and security of file transfer between user and portal site.
- The site must be a guarantee of the quality of the registered software. Quality insurance must be set up.
- The risk of competition with the private sector is limited. The experience shows that the open source applications and infrastructures open a new market for service companies and solution providers.

- Limiting the access of the POSS to the administrations only is not a major concern.

A concrete source of user requirement satisfaction may be found in investigating for best practices in European trans-border re-use of public sector software.

In this field the most interesting (and unique, since recent) example was found at the Danish National Labour Market Authority[58] (DNLMA) that decided to re-use a system developed in Sweden.
The DNLMA will migrate Swedish internet-based self-service-solutions for recruitment and job seeking.
In the beginning of April 2002, the DNLMA signed a contract with the IT-firm AU System, that had developed most parts of the Swedish system (*EU Tender no. 2001/s 188-129322*).
The developing phase began in the course of April 2002.
The two administrations are now collaborating together with the developers in order to transfer knowledge of the systems and the organisational aspects.
The DNLMA uses many functionalities "as-is" but also adds specific functionalities when needed.
In the future the two countries plan to "co-develop" the systems.
The DNLMA expects to save several million Euros by adapting the Swedish solution.
Another common project concerning the development of a new job bank has already started. The benefits of this project will be the division of costs and the possibility of trans-national functionality.

The various points of user requirements are detailed hereafter.

Quality of the OSS

The Development Life Cycle of an open source software (see chapter 1) is one of the major guarantees of its quality.
Yet, we could also ask feedback from effective POSS users with the following process : all POSS users can be requested to give an assessment ranging from zero (very bad) to ten (excellent). The average assessment and the total number of assessments could be made available in the documentation of the software on the POSS.
As done on general-purpose distribution sites, a software ranking according to the user assessment is required.

Contents of the POSS Portal Site

The POSS Portal site will give easy and user-friendly access to the following contents:

- Software items, i.e. applications that can be downloaded directly from the POSS Portal;
- Links to other sites providing interesting information on software pooling;

[58] Person of contact: Ghita Thiesen, GTH@ams.dk

- A Library of documents about OSS;
- A "News" page.

The Software

All the open source software that can be found on the POSS Portal have been used successfully by at least one administration.
The term "software" includes:

- Binary software distribution
- Software source code
- Code documentation (comment)
- Technical documentation
- User manual
- Administrator manual

The site will provide:

- A home page for each software item, which contains the descriptive form of the software;
- A searchable index, providing access to the home page of the software item.

The Software Description Form

All software items that will be available on the POSS portal will have a "descriptive form", containing the following information, as in the first IDA study on the use of open source software in the public sector (OSSPS)[59] :

- *A Title* (the name of the software package)
- *The Home Page* location, the address of the local page if hosted on the POSS Portal site
- *A short* and a *long description* of the software. The short description will be used for keyword search, which will be more efficient if the text is shorter.
- *The Category* of the software. Categories will be the same as in the OSSPS[59] study.
- *Information* regarding the *provider/main author* (contact person for the index entry)
- *Information* about the *support* organisation (e-mail address)
- *Size* of the software (in kilobytes and in number of lines of code)
- *Development Activity levels*, measured from mailing lists activity and/or frequency of releases
- The *target audience*:
  - Developers
  - End Users/Desktop
  - System administrators
  - Others
- The latest version number

[59] IDA study into the use of Open source Software in the Public Sector (http://europa.eu.int/ISPO/ida/)

- A link to change log/status information on the latest version
- A local copy of the latest software release (a single file) if no FTP server is available at the provider.
- References to papers or work describing or evaluating the software
- Key users of the software
- Development status.
    - Analysis: specifications available.
    - Alpha version: new development in use on pilot sites only.
    - Beta version: working version that still has to be widely used and/or tested.
    - Stable: in production and thoroughly tested.
    - Mature: stable, almost bug free, thoroughly tested and known to work in many different environments.
- License type
- Dependencies on proprietary intellectual property (Patents, proprietary tools required to use and/or modify the software)
- The *operating system* (for example, SourceForge[60] proposes multiple choice with 8 operating systems).
- The *programming language*(s) (for example, Sourceforge[60] propose(s) a multiple choice form with 45 languages)
- The *user language(s)* used for the user interface, the error messages… And an assessment (1 to 3) of the adaptability to other languages.

The Software Index

The Software Index will provide easy access to the home page of the software item, which contains the "promotional" information on its usage.

The software index will contain the fields described in the Software Description Form and will help users find open source software.
From the index, users will be able to access the software download areas.
For download, "Open source software" includes both code (binary distribution and sources) and documentation available under an open source (http://www.opensource.org) or an open content (http://www.opencontent.org) license.

The number of descriptive fields implemented is significantly larger than other publicly available databases for retrieving open source software such as Sourceforge[61].
The goal is to offer additional tools and information that can be used to evaluate the quality, reliability and suitability of the software in the index for administrations.

[60] SourceForge is the leading collaborative software development (CSD) platform, already used by more than a quarter million software developers worldwide. SourceForge is now available as an enterprise software product that helps companies accelerate development and centralize knowledge by improving internal visibility and control. SourceForge 2.0 is available as open source software at http://sourceforge.net/projects/alexandria
[61] http://sourceforge.net

A key feature of this index is the emphasis on attaching the "evidence" for the use of the software in administration. Such evidence consists of references to academic papers evaluating the software, key reference site and users, lists of certifications that the software has passed, regions where it is approved and deployed, and statistics on the size and activity of the resource.

Because the index is multilingual, and in order to minimise the burden of keeping the index up to date both for the portal site administrators and the project teams, the index will link to status information, rather than contain it. Most active projects already have this information available and it would be inconvenient to maintain it in two places.

The index can also contain direct links to projects hosted by similar sites (SourceForge[61], Inria[62]…)

Links

Submitting links (URLs) to interesting and relevant web information is a popular and useful activity in the Internet community. Although these links can be found using search engines, a specific tool for recording and searching targeted links will be more efficient to organize and keep the links up-to-date. The portal site may provide a searchable index of links of interest for the OSS Community even if these links are not specifically related to specific public sector software. The minimal information that should be recorded for each link is the following:

- Entry date
- E-mail address of submitter
- Link headline
- URL of link
- Short description of the interest of the link
- Language of the link
- Category (e.g.: Business case, user story, etc.)
- Date of last changes to the link contents
- Date of last verification of the validity of the link

Recently submitted links could be put on the POSS home page and could be the subject of news or broadcasts from the POSS and other portal sites.
Here are some examples of interesting links :

- General purpose Open source Products links (Linux, FreeBSD, popular OSS tools…).
- Links to Specialised National agencies (Atica[63], BerliOS[64]).
- Links to general purpose existing POSS services (SourceForge.net[61], Freshmeat.net[65]…).
- Links to Open source Organisations (Opensource.org[66]).
- Specialised sites, as Euspirit.org[67].

[62] http://www.inria.fr/valorisation/logiciels/index.fr.html
[63] http://www.atica.pm.gouv.fr/
[64] http://www.berlios.de/
[65] http://freshmeat.net/
[66] http://www.opensource.org/
[67] http://www.euspirit.org/

Library

The information needed to raise awareness of open source software, to understand its use, to acquire and to support the software, is scattered throughout the Internet: this is one of the major barriers to accelerating the uptake of open source applications. Therefore we recommend the creation of a library where articles, white papers, market statistics and case studies can be found to assist in education and to provide resources for the development of plans and business cases. The intent is to have a limited but high quality collection of information.

The library should focus on:

- Articles related to open source deployment in administrations;
- Suggested policy and best practices for open source adoption in administrations;
- Legal information and other requirements for deploying open source software in European countries (e.g.: Privacy rules, security requirements, required certifications, important payers, etc.)

The index should contain at least:

- Entry date
- Title
- Language
- Summary

News

The OSS Community members need to be kept up-to-date both with changes in the field of open source Software and with the activity in the community. Individual community members should have access to a news page that would contain information about OSS like for instance the announcement of events, meetings, discussion forums, new software releases, …

An index entry should include:

- Title
- Category. For example :
- Release
- Meeting
- Event
- General news
- Submitter name
- Entry date
- Description

Creating a Community

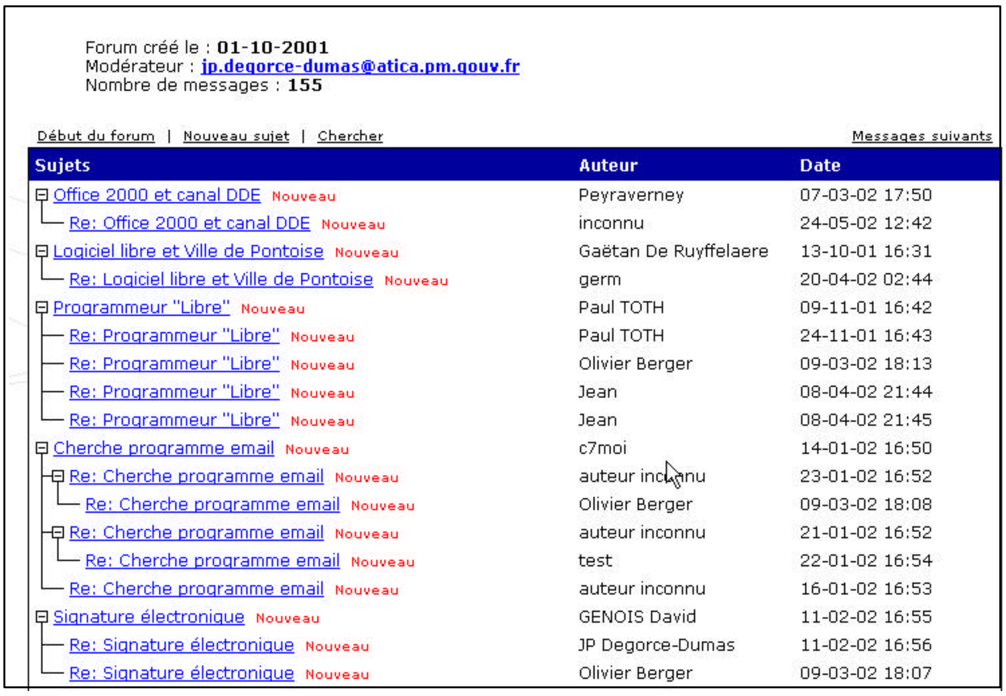
The potential users of the POSS Portal site will share the same needs and have some common tools. In order to help them improve their work methods and to benefit from each other's experience, the POSS site must provide users with tools to reinforce the exchanges within the OSS Community : the POSS users must be able to post questions, share information, and exchange comments. They need a virtual meeting place to reinforce the feeling of belonging to a real Community.

Forums

Aim

Forums are intensively used in the open source community.
Members can join forums and post questions on them. All the forum subscribers can then see these questions. Subscribers can answer this question, or just bring an element to the discussion. The history (questions and answers) of each topic is visible for all the subscribers in HTML format.

Figure shows an example of forum layout (French Prime Minister Service ATICA).



Forum créé le : **01-10-2001**
Modérateur : **jp.degorce-dumas@atica.pm.gouv.fr**
Nombre de messages : **155**

Début du forum | Nouveau sujet | Chercher   Messages suivants

Sujets	Auteur	Date
Office 2000 et canal DDE Nouveau	Peyraverney	07-03-02 17:50
Re: Office 2000 et canal DDE Nouveau	inconnu	24-05-02 12:42
Logiciel libre et Ville de Pontoise Nouveau	Gaëtan De Ruyffelaere	13-10-01 16:31
Re: Logiciel libre et Ville de Pontoise Nouveau	germ	20-04-02 02:44
Programmeur "Libre" Nouveau	Paul TOTH	09-11-01 16:42
Re: Programmeur "Libre" Nouveau	Paul TOTH	24-11-01 16:43
Re: Programmeur "Libre" Nouveau	Olivier Berger	09-03-02 18:13
Re: Programmeur "Libre" Nouveau	Jean	08-04-02 21:44
Re: Programmeur "Libre" Nouveau	Jean	08-04-02 21:45
Cherche programme email Nouveau	c7moi	14-01-02 16:50
Re: Cherche programme email Nouveau	auteur inconnu	23-01-02 16:52
Re: Cherche programme email Nouveau	Olivier Berger	09-03-02 18:08
Re: Cherche programme email Nouveau	auteur inconnu	21-01-02 16:52
Re: Cherche programme email Nouveau	test	22-01-02 16:54
Re: Cherche programme email Nouveau	auteur inconnu	16-01-02 16:53
Signature électronique Nouveau	GENOIS David	11-02-02 16:55
Re: Signature électronique Nouveau	JP Degorce-Dumas	11-02-02 16:56
Re: Signature électronique Nouveau	Olivier Berger	09-03-02 18:07

Figure 10 Forum layout example

In this way Forums provide a place where people can help each other and share useful information.

Functionalities

The portal site must provide functionalities to initiate forums.

Forums can be public or private (authorised users only).

Topics

There should be general forums about open source, administration… and software item specific forums.

Implementation

Forums require a connection to the POSS site to access the information.

Mailing lists

Mailing lists allow addressing individual people directly, via their e-mail address, whereas the Forums require a connection to the POSS portal site.
By enrolling to a public mailing list, the community members will be able to discuss ongoing activities for specific software and to receive the information exchanged on this software. This gives the necessary transparency to the work, and is an ideal tool to incorporate feedback and guidance from the community.

Promoting the Use of the POSS Portal Site

As we have existing examples of POSS-like sites in production, we know that the POSS Portal site is a feasible project. Its success will depend on the quality of its contents and of its organisation. Yet, this is not sufficient: in order to be a success, the POSS Portal sites must have a minimum number of visits. Therefore, potential users must be informed of the portal's capabilities. A prerequisite for satisfying this information requirement is to reach a large number of potential participants, disseminating information that reaches different profiles (roles). In particular, the objectives of dissemination are:

- To develop a critical mass of participants to ensure long-term self-sustainability and evolution of open solutions in administrations. When initiating a new project of software development inside a European administration, the project manager and developers should think first of the POSS portal site;
- To initiate "Best Practices" communities for open source applications in European administrations;
- To promote the development of common solutions for several administrations.

Initiate key communities for applications

At the moment of writing, each administration has its own development methods and no project implies participation beyond the originating administration. Few public organisations are aware of developments implemented in other administrations, even if those developments would satisfy the needs of their own organisation.
One of the goals of the POSS project is to create and disseminate information to help project communities grow and thrive.

The successful formation of a community depends on the following required conditions:

- A meeting place that enables communication between administrations
- A core group of “players” that initiates activities

Particularly in the set-up phase of a community, a core group of administrations (leading countries) has to produce a highly visible activity (software registration, advising…) in the POSS community. This group has to make an explicit effort until a sufficient contribution from other administrations will guarantee the necessary minimal activity on the POSS Portal site. It is important that this group reaches out to new candidate members with complementary and diverse skills. Getting to the point of self-sustaining project momentum requires more than sheer volume of participants, but also needs some specialisation of participants in different community roles.
We suggest launching the project with a small database of software components collected in leading European administrations.

Developing the Critical Mass

To develop a critical mass of participants, a critical number of administrations must become part of the community. Therefore, some conditions need to be satisfied:

- People must be aware of the capabilities of the OSS Community and willing to know more about the use they can make of it for their own needs.
- People must be convinced of the benefits of an active participation in the POSS community.
- Promoters of the POSS must be identified in the administrations.

To increase the awareness, multiple media are used to disseminate relatively simple messages that awake the interest of potential participants.
Each message is directed to a specific group, customised for the targeted audience and optimised for the media used. It pushes forward the benefits of being part of the POSS community and points at ways to get more information.

To convince interested parties to join the POSS community, additional communication is necessary. The communication is much more effective when the message can be personalised to the specific interests and needs of the candidate participants, and strong evidence for benefits can be conveyed. Usually, a mixture of the following kinds of communications is adequate in this process:

- Information about the benefits of participation
  - Available in increasing level of detail
  - Available for different interests and backgrounds
  - Possibly available on multiple media
- Evidence for the benefits in the form of:

- Recommendations from trusted sources, peer reviews or other administrations;
- Published success stories;
- Lists of successful members (or member organisations);
- Contacts with involved administrations.

Yet, to guarantee the success of the POSS project, it will need active promoters in the administrations. These promoters must have the full support of their hierarchy in order to dedicate some time to the set-up and the launch of the POSS site in their role of POSS users.

POSS Portal Site services

The portal site provides a set of services to its users. These services and the community response to them can make the portal a warm, inviting, friendly place where valuable contacts can be made, information obtained and work done.
Services must be selected and designed based on the following criteria:

- The service must be sustainable. The approach to sustainability is to provide services that are highly automated. Services must require little ongoing support in order to avoid putting in place a full-time maintenance team.
- The services must respect linguistic diversity but this characteristic is high costly.
- Automated usage statistics and surveys must be put in place to provide ongoing feedback.
- The services must remain simple, easy to use and well documented. The major problem of most similar tools is their complexity and lack of documentation.

The portal site must be highly automated (user registration, links addition…). Yet, this will not totally suppress the systems administration: it is not a good idea to allow anyone to create users, register software, create forums… Supervision will guarantee the quality and the reliability of the site.

The services must also be secured. Security is characterised as the preservation of:

- *Confidentiality:* ensuring that information is accessible only to authorised users;
- *Integrity:*
  - Safeguarding the accuracy and completeness of information ;
  - Processing methods.
- *Availability:* ensuring that authorised users have access to information and associated assets when required.

One of the most important security characteristics will be the encryption of the files that will be transferred from and to the POSS.
This applies to:

- The members who want to download software from the site
- The members who have to upload files on the POSS site server.

A typical private-public key crypto system must be put in place with use of PKI certificates. This system is described in chapter 4.

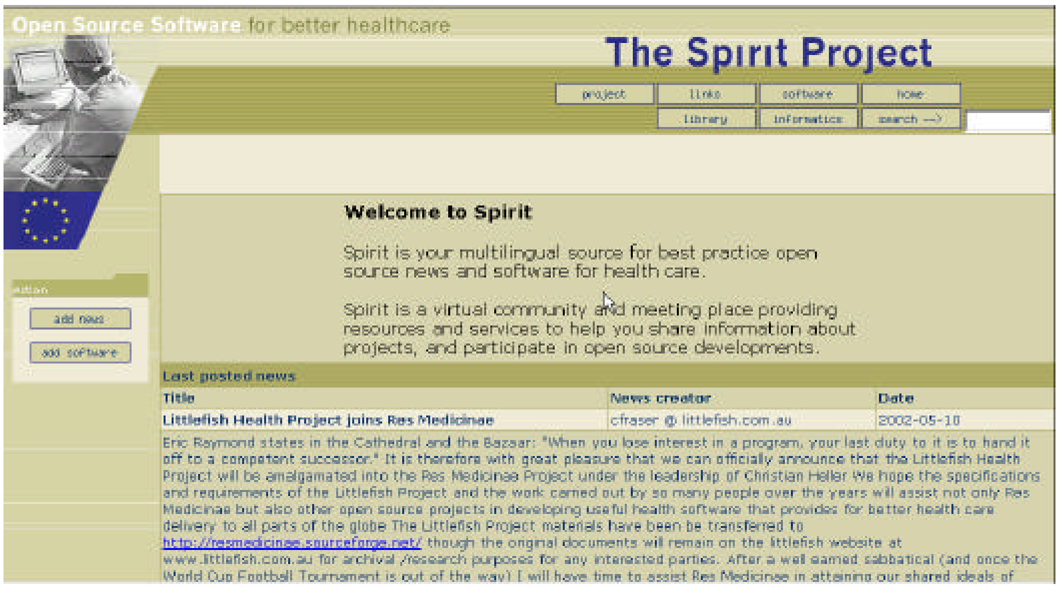
Home Page

Description

The Home Page is the main entrance point of the POSS portal site, and the main publicised address for the site.
The main requirements for the initial home page of the POSS portal site are:

- Provide immediate access to basic background information on the project, project status, and participants;
- Provide a site map;
- Provide an easy and attractive entrance into the POSS portal page;
- Display the availability of the various languages in which the portal is accessible, and allow for easy switching between languages;

Here are two examples of home page for sharing software portal.
Figure 11 shows the home page of the Spirit project[67] (open source sharing for health care).



Welcome to Spirit

Spirit is your multilingual source for best practice open source news and software for health care.

Spirit is a virtual community and meeting place providing resources and services to help you share information about projects, and participate in open source developments.

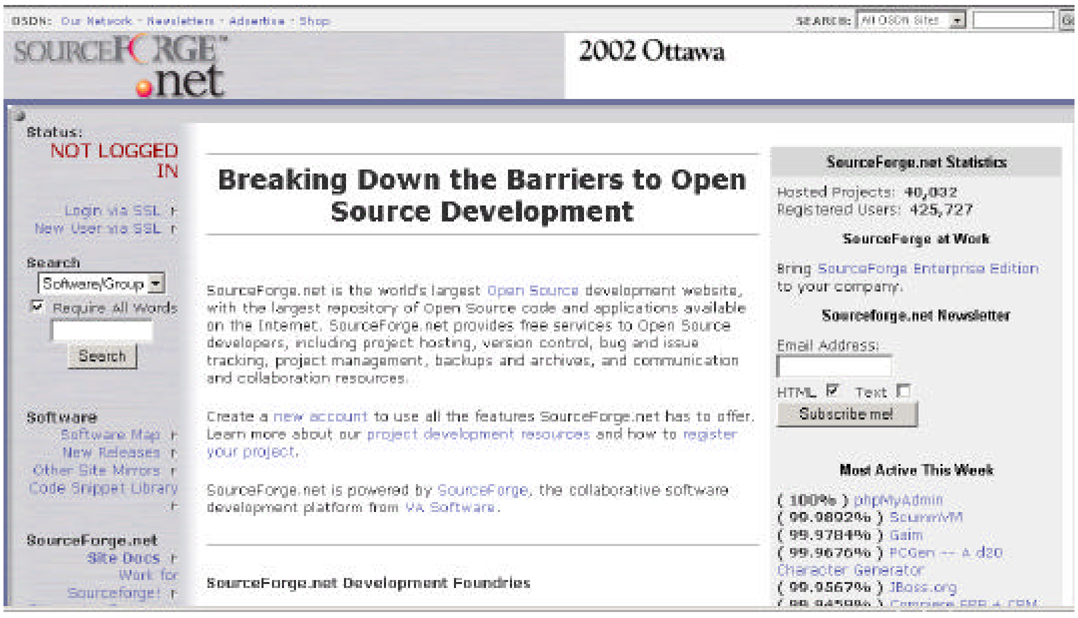
Last posted news

Title	News creator	Date
Littlefish Health Project joins Res Medicinae	cfraser @ littlefish.com.au	2002-05-10

Eric Raymond states in the Cathedral and the Bazaar: "When you lose interest in a program, your last duty to it is to hand it off to a competent successor." It is therefore with great pleasure that we can officially announce that the Littlefish Health Project will be amalgamated into the Res Medicinae Project under the leadership of Christian Heller We hope the specifications and requirements of the Littlefish Project and the work carried out by so many people over the years will assist not only Res Medicinae but also other open source projects in developing useful health software that provides for better health care delivery to all parts of the globe The Littlefish Project materials have been be transferred to http://resmedicinae.sourceforge.net/ though the original documents will remain on the littlefish website at www.littlefish.com.au for archival /research purposes for any interested parties. After a well earned sabbatical (and once the World Cup Football Tournament is out of the way) I will have time to assist Res Medicinae in attaining our shared ideals of

Figure 11 Spirit project home page

Figure 12 shows the SourceForge[61] home page.


OSDN: Our Network · Newsletters · Advertise · Shop

SEARCH: All OSDN Sites

2002 Ottawa

Status:
NOT LOGGED IN

Login via SSL
New User via SSL

Search
Software/Group
☑ Require All Words
Search

Software
Software Map
New Releases
Other Site Mirrors
Code Snippet Library

SourceForge.net
Site Docs
Work for Sourceforge!

Breaking Down the Barriers to Open Source Development

SourceForge.net is the world's largest Open Source development website, with the largest repository of Open Source code and applications available on the Internet. SourceForge.net provides free services to Open Source developers, including project hosting, version control, bug and issue tracking, project management, backups and archives, and communication and collaboration resources.

Create a new account to use all the features SourceForge.net has to offer. Learn more about our project development resources and how to register your project.

SourceForge.net is powered by SourceForge, the collaborative software development platform from VA Software.

SourceForge.net Development Foundries

SourceForge.net Statistics

Hosted Projects: 40,032
Registered Users: 425,727

SourceForge at Work

Bring SourceForge Enterprise Edition to your company.

Sourceforge.net Newsletter

Email Address:
HTML ☑ Text ☐
Subscribe me!

Most Active This Week

(100%) phpMyAdmin
(99.9892%) ScummVM
(99.9784%) Gaim
(99.9676%) PCGen -- A d20 Character Generator
(99.9567%) JBoss.org

Figure 12 SourceForge home page

Objectives

The home page must provide a clear and easy access to the services offered by the POSS portal site.

Members[68]

Description

People interested in joining the community must follow the registration process.
The first step to take is the identification of the role. The POSS portal site must then provide clear information about the various roles so that the future member can clearly define his role (see chapter 5) in the POSS.
Afterwards, he can fill out the corresponding registration form on the web. A user name and a password must be chosen.
The registration form is then submitted to the POSS and the future member receives a registration mail. He has to reply to the mail for confirmation. After reception and acceptance of the registration confirmation by the POSS, the member is informed and has access to his profile functionalities.

Figure 13 shows the registration process.

[68] The word "***member***" refers to any individual who can be registered on the POSS whatever his role.

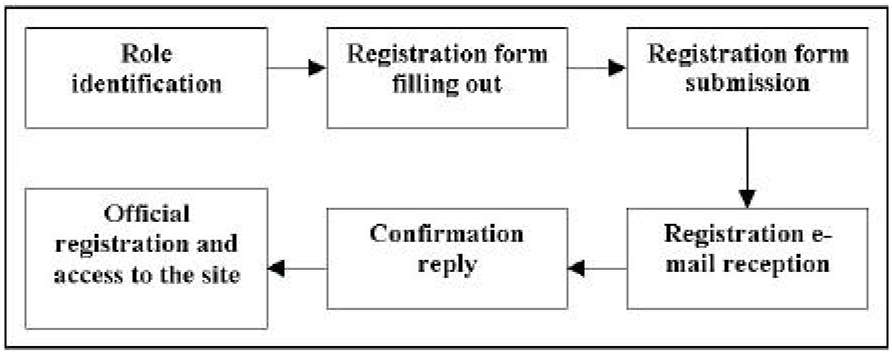

Role identification
Registration form filling out
Registration form submission
Registration e-mail reception
Confirmation reply
Official registration and access to the site

Figure 13 Member registration process

Objectives

The registration will allow members to have access to the selected functionalities of the system corresponding to their profile. The registration process must be as simple as possible in order to limit the burden for future members.

Software

Description

The services for the software include:

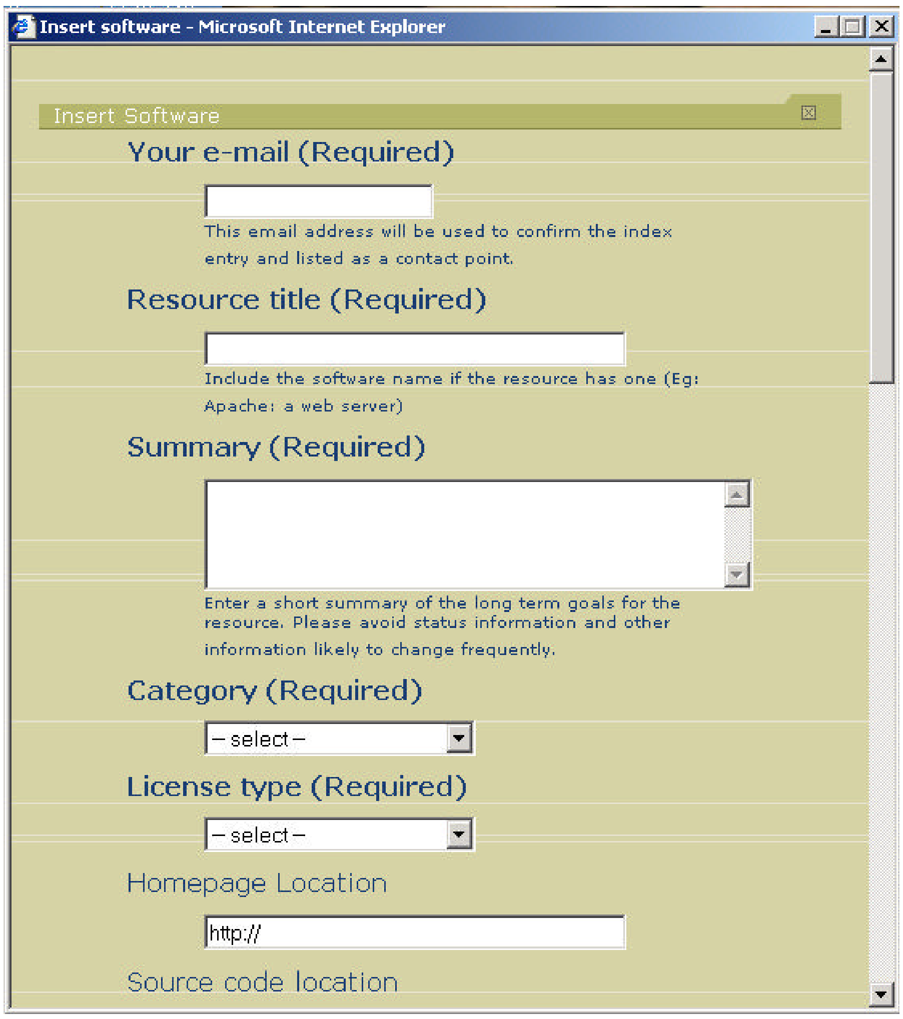
- Software submission
- Software modification
- Software download

Software submission

The first step for the member is to identify software that can be shared. The candidate software must then be prepared: it must answer POSS requirements for the code and the documentation (e.g. UML, naming conventions). Afterwards, the member submits the candidate software.
The following elements must be submitted:

- Binary software distribution
- Software source code
- Code documentation (comment)
- Technical documentation (functionalities)
- User manual
- Administrator manual
- Test plan

The POSS portal site must then provide a web-based software submission form. The member must enter the descriptive fields listed above, the name of the software responsible and possibly, the names of the software developers. The submission of this form includes the acceptance of the contract between the POSS and the submitter about the redistribution of the software. As example, Figure 14 shows the Spirit software submission form.



Insert software - Microsoft Internet Explorer

Insert Software

Your e-mail (Required)

This email address will be used to confirm the index entry and listed as a contact point.

Resource title (Required)

Include the software name if the resource has one (Eg: Apache: a web server)

Summary (Required)

Enter a short summary of the long term goals for the resource. Please avoid status information and other information likely to change frequently.

Category (Required)

– select –

License type (Required)

– select –

Homepage Location

http://

Source code location

Figure 14 Spirit software submission form

The user must also upload the software files for acceptance.
Once the software is accepted, an entry is then added in the software index and the software responsible receives the rights to administrate his software area (software files, HTML software home page, developers management…).
If the developers don't have facilities to host HTML description pages and software files, the POSS portal site must provide it.
Disk space must be available to store HTML pages describing the software component accessible from the software index.
Disk space must also be available on the FTP server to store software files (binary distribution, sources and documentation) and allow download by users.

A forum and a mailing list dedicated to the software are also created by the POSS and are administrated by the software responsible.

Figure 1 shows the software submission process for members.

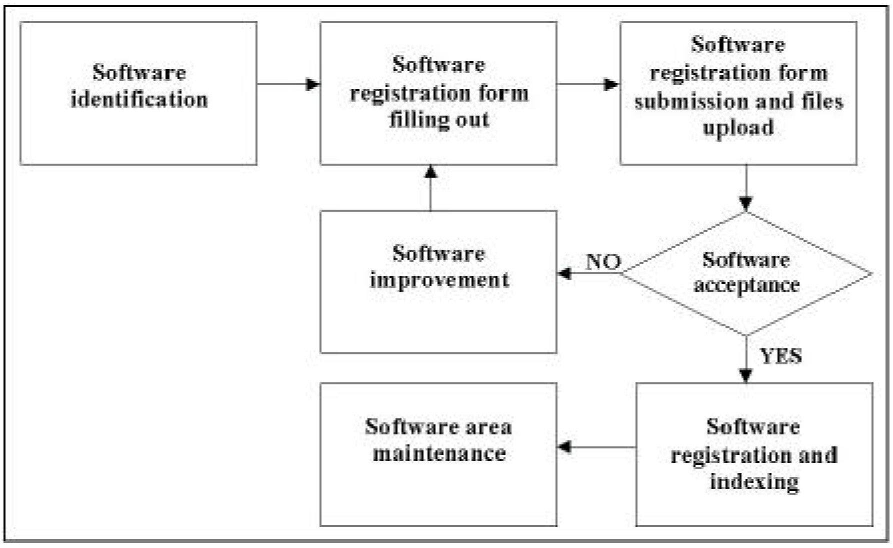

Software identification
Software registration form filling out
Software registration form submission and files upload
Software improvement
NO
Software acceptance
YES
Software area maintenance
Software registration and indexing

Figure 1 Member software submission process

Software modification

Each software modification must be accepted before changing the index.
The POSS portal site must provide a software modification form that can be submitted. After acceptance of the software modification, the index is updated and the versioning system takes care of the version number.
The POSS must provide facilities to upload files.

Figure 2 shows the software modification process.

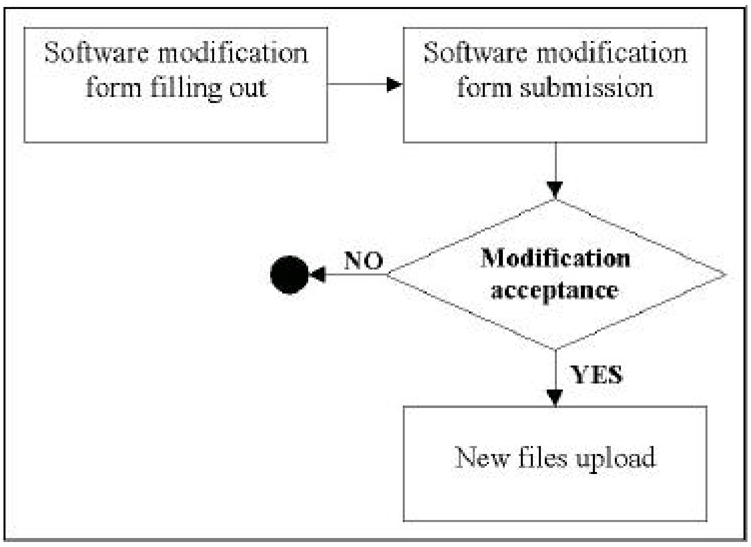

Software modification form filling out
Software modification form submission
NO
Modification acceptance
YES
New files upload

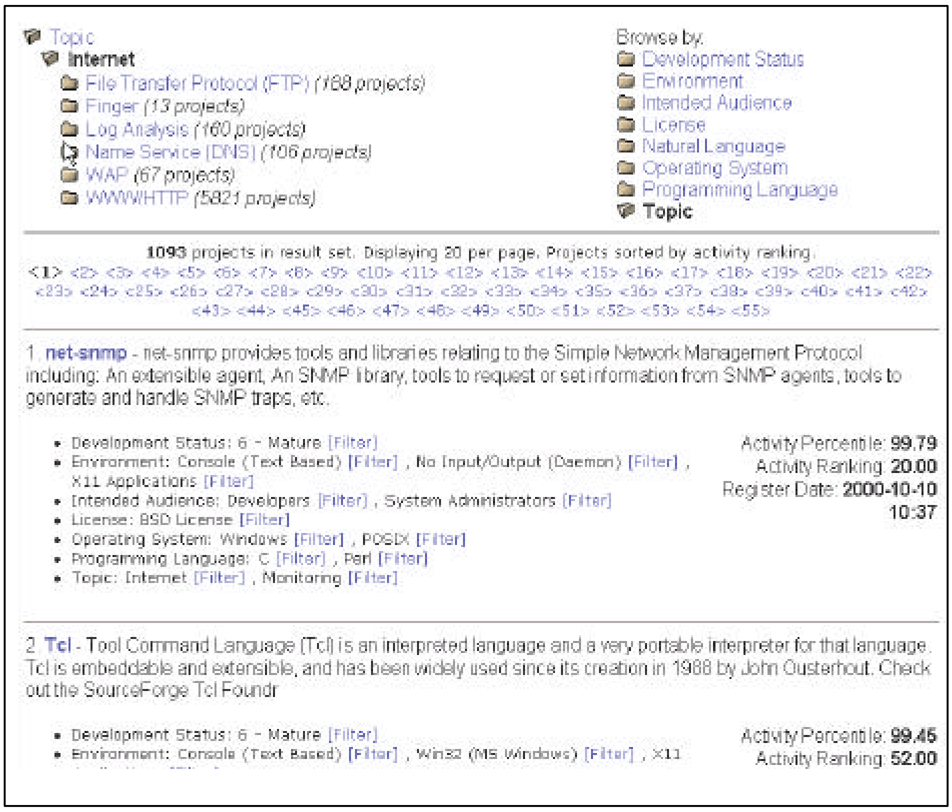
Figure 2 Software modification process

Software download

The first step to take for software downloading is to perform a search in the software index. Users must be able to search the index using any of the descriptive fields. It will also be possible to browse projects according to each field with predefined multiple choices:

- Category
- Intended audience
- Development status
- License
- Operating system
- Programming language
- User language

Figure 3 shows an example of software browsing on SourceForge.


Topic
Internet
File Transfer Protocol (FTP) *(168 projects)*
Finger *(13 projects)*
Log Analysis *(160 projects)*
Name Service [DNS] *(106 projects)*
WAP *(67 projects)*
WWW/HTTP *(5821 projects)*

Browse by:
Development Status
Environment
Intended Audience
License
Natural Language
Operating System
Programming Language
Topic

1093 projects in result set. Displaying 20 per page. Projects sorted by activity ranking.
<1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14> <15> <16> <17> <18> <19> <20> <21> <22> <23> <24> <25> <26> <27> <28> <29> <30> <31> <32> <33> <34> <35> <36> <37> <38> <39> <40> <41> <42> <43> <44> <45> <46> <47> <48> <49> <50> <51> <52> <53> <54> <55>

1. **net-snmp** - net-snmp provides tools and libraries relating to the Simple Network Management Protocol including: An extensible agent, An SNMP library, tools to request or set information from SNMP agents, tools to generate and handle SNMP traps, etc.

- Development Status: 6 - Mature [Filter]
- Environment: Console (Text Based) [Filter] , No Input/Output (Daemon) [Filter] , X11 Applications [Filter]
- Intended Audience: Developers [Filter] , System Administrators [Filter]
- License: BSD License [Filter]
- Operating System: Windows [Filter] , POSIX [Filter]
- Programming Language: C [Filter] , Perl [Filter]
- Topic: Internet [Filter] , Monitoring [Filter]

Activity Percentile: **99.79**
Activity Ranking: **20.00**
Register Date: **2000-10-10 10:37**

2. **Tcl** - Tool Command Language (Tcl) is an interpreted language and a very portable interpreter for that language. Tcl is embeddable and extensible, and has been widely used since its creation in 1988 by John Ousterhout. Check out the SourceForge Tcl Foundr

- Development Status: 6 - Mature [Filter]
- Environment: Console (Text Based) [Filter] , Win32 (MS Windows) [Filter] , X11

Activity Percentile: **99.45**
Activity Ranking: **52.00**

Figure 3 Software browsing on SourceForge

The POSS portal site must provide a search form for the software index.

If several software items are returned by the search, a choice must be made by asking additional information through forums, mailing lists or directly to the software responsible. To help members to choose software, the POSS should provide statistics and information about each project: download and activity rate, rating from other users…

Once the desired software is identified, the download contract must be accepted. The POSS must then provide a contract acceptance mechanism. The software can then be downloaded, either from the POSS server or from the provider server.
The software can possibly be adapted for the new user needs and is finally deployed.

Support for software deployment can be obtained from forums, mailing lists or directly from the software responsible or developers.

Figure 4 shows the software download and deployment process.

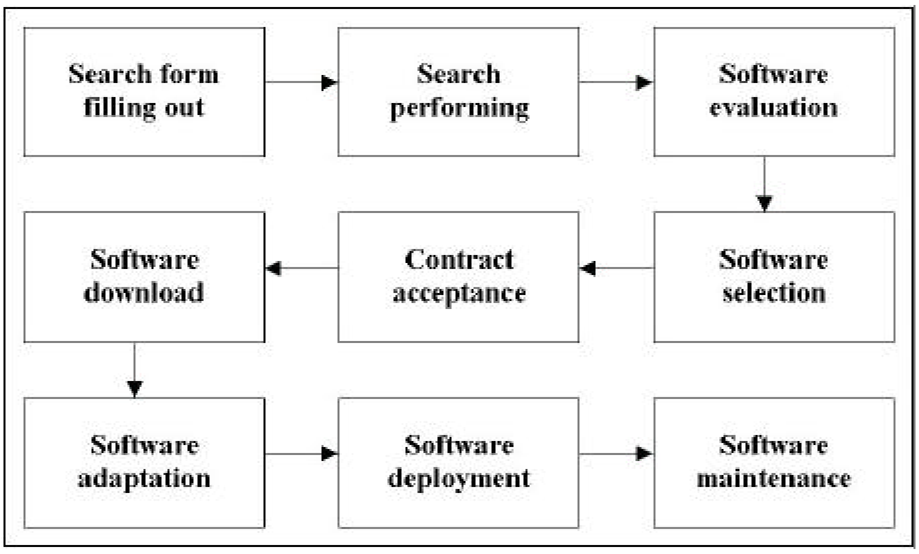


Figure 4 Software download and deployment process

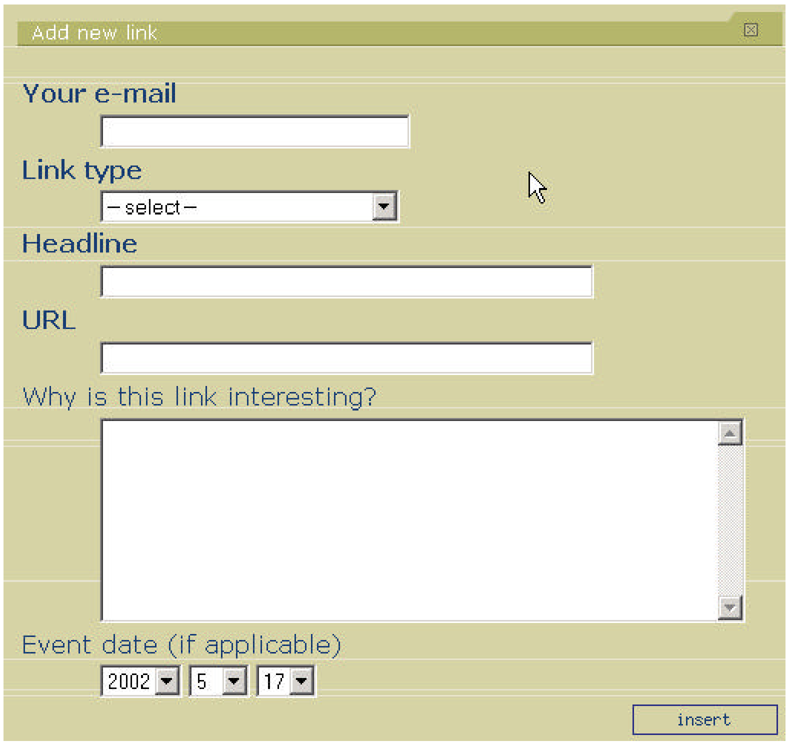
Objectives

Software services aim at providing easy access to software of quality responding to the needs of the POSS users.

Links

Description

The POSS portal site must provide an “add link form” to allow members to add links dynamically.
Figure 5 shows the “add link form” on Spirit[67].



Your e-mail	
Link type	– select –
Headline	
URL	
Why is this link interesting?	
Event date (if applicable)	2002 5 17

Figure 5 Add link form on Spirit

Forms must also be available to search the links index with key words. Members should also be able to browse links according to

- Language
- Category
- Date of submission (range)

Objectives

Members must be able to access interesting links related to the POSS portal site and add links that can be of interest to other members.

Library

Description

The POSS portal site must provide the capability to present new material that should be added to the library.

Members must also be able to search an annotated index and then download a copy of a document directly or follow a link to a referenced text.

The library should contain texts of interest from the start.
Afterwards, members can submit texts whose contents will be checked before registration in the library.

Objectives

The library aims at providing valuable and up-to-date documentation on the development, maintenance, implementation and deployment of shared software and open source software.

News

Description

The POSS portal site must provide an “add news form” to allow members to add a piece of news dynamically.
Forms must also be available to search the news index with key words.

Members should also be able to browse news according to

- Category
- Entry date (range)
- Submitter name

Objective

News aims at providing valuable and up-to-date news on the development, maintenance, implementation and deployment of shared software and open source software.

Forums

Description

The POSS portal site must provide a forum submission form. Registered members can submit forum topics to the POSS. Forums can be public or of limited access.
If the forum is accepted by the POSS, the submitter receives the rights to manage and moderate it.

Figure 20 shows the forum submission process.

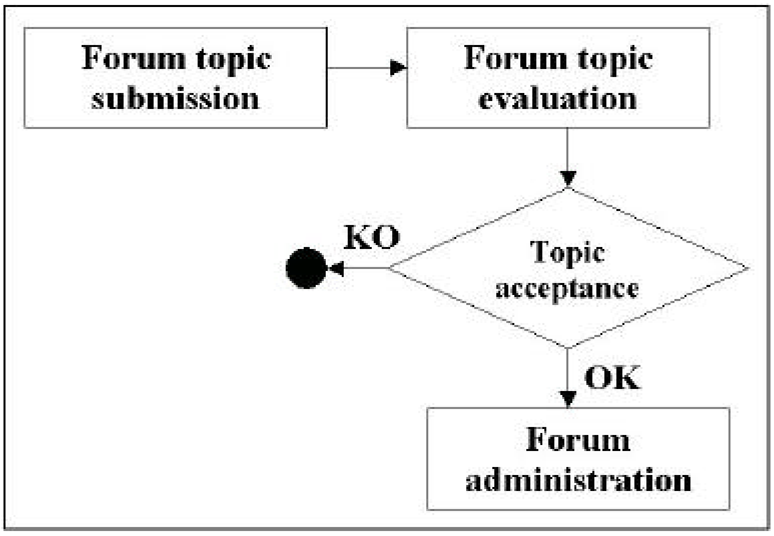


Figure 20 Forum submission process

Objective

Forums aim at providing the meeting places required by the POSS users to exchange information.

Mailing list

Description

Three types of mailing lists will be available:

- General Mailing List
  The General Mailing List contains the e-mail address of all the people registered on the POSS Portal Site. It can only be used by the system administrators to broadcast messages in case of upcoming events, potential problems, etc …
- Public Mailing List
  Each project responsible should also be able to initiate mailing lists relative to his project. The community members would be able to discuss ongoing project activities. This would give the necessary transparency to the work, and would be an ideal tool to incorporate feedback and guidance from the community.
- Private Mailing Lists
  POSS users can create, use and manage their own mailing lists.

The POSS portal site must provide a mailing list request form allowing members to request the creation of a mailing list. After the creation, the submitter receives the rights to manage his mailing list.
Members must be able to

- Create/Edit/Delete their own mailing list
- Send mail to existing lists

- Enrol in general and public lists
- Cancel own enrolment in general and public list

Objective

Mailing lists provide the ability to send messages to a targeted audience.

Opinion surveys

Description

Users' opinion about the site will be collected through surveys and questionnaires.
From any location on the site, it must be possible to reach the survey page. In this area, members will be able to submit suggestions about the site (new functionalities, changes…) and fill out a monthly opinion survey about the site services.

Objective

The survey must aim at understanding the users' needs and improving the functionalities of the POSS.

Service providers list

Description

It may be interesting to find on the POSS a list of enterprises delivering various types of services (consultancy, support, integration, distribution) in Open source (not exclusively according to the neutrality principle).

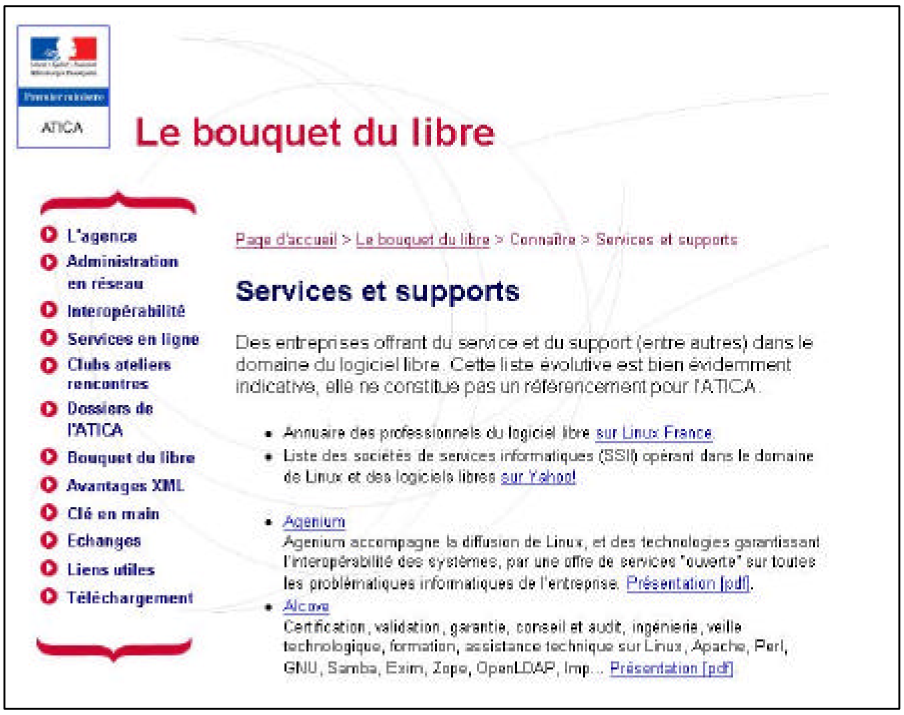
The acceptance of this service has to be tested (as it should be maintained by the enterprises and cannot engage POSS liability).

Service providers should then receive a specific role and ID inside the POSS and a corresponding application should allow them to :

- create
- update, delete
- attach links to presentation documents

in the corresponding service provider database.

POSS users must be able to consult the list, browse them by country and type of service



Le bouquet du libre

Page d'accueil > Le bouquet du libre > Connaître > Services et supports

Services et supports

Des entreprises offrant du service et du support (entre autres) dans le domaine du logiciel libre. Cette liste évolutive est bien évidemment indicative, elle ne constitue pas un référencement pour l'ATICA.

- Annuaire des professionnels du logiciel libre sur Linux France.
- Liste des sociétés de services informatiques (SSII) opérant dans le domaine de Linux et des logiciels libres sur Yahoo!

- Agenium
  Agenium accompagne la diffusion de Linux, et des technologies garantissant l'interopérabilité des systèmes, par une offre de services "ouverte" sur toutes les problématiques informatiques de l'entreprise. Présentation [pdf].
- Alcove
  Certification, validation, garantie, conseil et audit, ingénierie, veille technologique, formation, assistance technique sur Linux, Apache, Perl, GNU, Samba, Exim, Zope, OpenLDAP, Imp... Présentation [pdf].

Figure 21 Services and support by Atica

In France, the official Prime Minister ATICA site illustrates this cooperation with the private sector.

Objective

Service provider identification corresponds to a user need to find support in his own country area.
It illustrates the complementary (non competition) between the knowledge sharing service and the industry.
Later, it may be a source of income for the POSS.

4. Technical Design Framework

Objectives

In the hypothesis a Pooling Open source Software service should be set up, the present study is also a feasibility study : it must provide guidelines on the POSS technical framework and standards, together with illustrations of such possible architecture (similar services and recommendations).

To illustrate the possible technical design, we selected hereafter a list of software components that are - also – distributed as Open source. This list is not a rigid technical requirement of the POSS service, and therefore other components may be proposed or used when justified at the time of constructing the POSS service (if such decision should be taken).

Topic 1 is a presentation of the technical framework of standards to respect when designing the POSS. This part describes the overall site design and recommended security features.

Topic 2 gives a list and a description of potential tools for the set up of the POSS. The list is made up of tools that are already in use at similar sites and that have proven their full compatibility.

Topic 3 describes the tasks that need to be fulfilled for the set up the POSS site. For each task, the responsibilities are identified.

The last section gives examples of similar existing tools and their components.

General site design

Navigation

Immediate and easy access to information is very important. The contents keep visitors coming to your site and motivate them to come back again. However, the best site contents are useless if your visitors cannot get to it. Quality contents and effective navigation belong together -- without one, the other is useless.

Clear and Consistent

Navigation should be clear and consistent. Primary topic categories and important links should be on every page, at the same position, and in the same sequence. For example (see Figure),

- the main topics accessible on the POSS site at the top of the page,
- general info and global functionalities, on the left,
- the link to the next screen at the bottom of the page

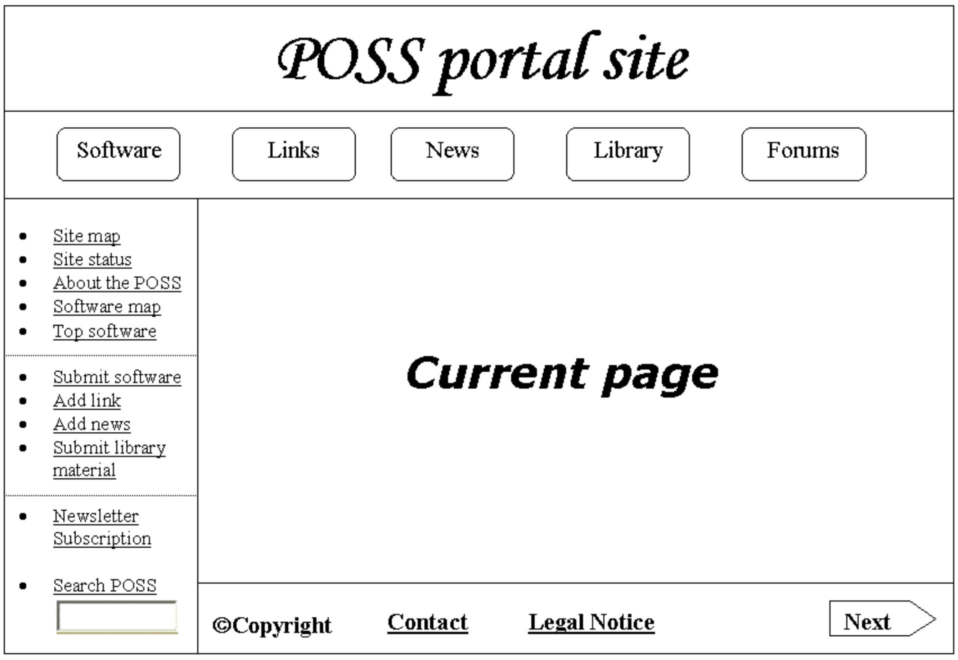


Figure 22 POSS portal site view

Locate primary links high enough on the page so that they are visible with no need to scroll.

Organized Site Structure

The key to effective navigation is good organization. Organize your site's contents into clear topic and sub-topic categories. For example:

- Projects
  - Submit
  - Search
  - Browse
  - Manage
- News
  - Search
  - Browse
  - View recent
  - Insert
- Links
  - Search
  - Browse
  - View recent
  - Insert
- Library
  - Search
  - Browse
  - Submit

In addition to links to primary topics and important information, links to sub-topics must be available, whenever possible, within each topic category.
Site map and site search functionalities can be of great help for users.

Generic template

To provide a user-friendly navigation interface supporting an efficient usage of the POSS, the look-and-feel and the access to its functionalities must be homogeneous throughout the site. Therefore, a generic template, defining the look-and-feel of all the pages of the site, must be designed from the start for all future implementations.
This generic template will settle the usage of colors, fonts, background and other presentation information.
The usage of a generic template is fundamental: it allows changing the look and feel of the whole site by simply modifying the global configuration of the site to point to another generic HTML template.

Multilingual features

The use of templates will also allow an easy management of multilingual pages.
Each page of the site will have a template sub-classed from the generic template (to inherit its global features). The contents will be included in an active way thanks to scripting language.

Figure 23 shows an example of a page template.
The page "***Page X***" has a template sub-classed from the generic template. This template defines its structure (order of the components) together with identifiers to be able to insert its contents in different languages. The contents of the "***Page X***" in the different languages are stored in a database and dynamically built-in in the template according to the user language at display time.

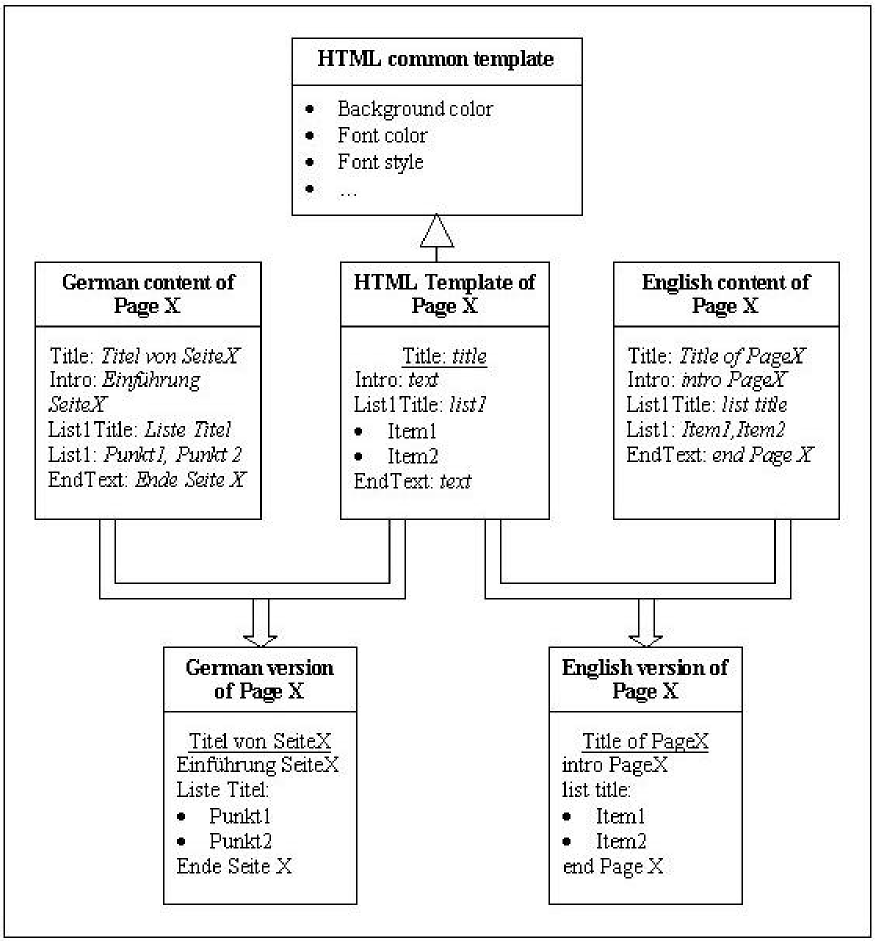


Figure 23 HTML template

Pages types

The structure of the site is subdivided into several distinct document pages, some static and some active (with data extracted from the database).

Static pages are used for the collection of slowly changing contents. For example, press materials (both official press releases and documents released to the press), generic documents about open source, administration IT, documents specific to the POSS project…

The active parts are the pages dealing with the database itself and providing active services.
Data to be displayed will be extracted directly from a database.
Dynamic pages can be developed with a scripting language allowing embedding SQL requests in HTML code.

Security features for files transfers

Security principles

All the file transfers from and to the POSS need to be secured. Files can be encrypted using digital certificates.
Security solutions using digital certificates rely on a public key cryptography in which each user has a pair of cryptographic keys: one private key that is kept private by the user, and one related public key made public.
A Digital Certificate is a digitally signed statement that certifies the binding between the owner’s identity information and his/her electronic public key. This certified public key can be used to encrypt confidential information to the certificate owner and/or to verify digital signatures generated by the certificate owner.
The certified public key is linked to the private key of the certificate owner in such a way that:

- ***A digital signature*** is computed from the message (file) using the private key of the signer. It is a small size coded file appended to the signed message. The verification of the digital signature uses the certified public key of the signer. If the result of the check is positive, the recipient can trust its origin and has the guarantee that nothing has been modified in the message since the signature process.
- ***Confidentiality*** is obtained from the ciphering of the message with the certified public key of the recipient. The only way to decrypt a ciphered message is to use the corresponding private key that is supposed to be known only to the certificate owner.

Digital certificates provide thus solid assurance that a public key actually belongs to the right entity whose identity has been certified by a Certification Authority, a known trusted third party, which controls and confirms the accuracy of the binding between a public key and its legitimate owner. This mechanism will be used intensively on the POSS.

POSS Security Objectives

On one hand, users need to securely download applications. Therefore, the POSS site must publish the public form of the certificates, enabling the users to verify the origin and the integrity of the downloaded applications with the technique of digital signature as explained above.

On the other hand, all the developers of the POSS who will upload files on the POSS server must have a certificate issued by a trusted authority certifying the developer's identity. The public certificates of POSS developers should be published on the POSS site.

Security set up

The security features can be set up using the following tools:

- Certification Authorities
- A secure, X.509 PKI based transfer (upload-download) system (see below the proposed tool)

The POSS must first generate private and public keys using special tools (such as keytool[69]), or a browser.
Then, a certificate must be asked to a Certification Authority.
There are many public Certification Authorities (CA) that provide PKI certificates (X.509) for personal identification, software signing, and vendor identification purposes.

Certificates from major certification authorities and/or national authorities must be accepted. The details of the CA policies and identification mechanisms will be linked from the POSS site so that users can verify for themselves their degree of trust in specific certificates.

[69] http://java.sun.com/products/jdk/1.2/docs/tooldocs/solaris/keytool.html

The framework of tools requested to be integrated to answer requirements.

Basic Tools

To construct the POSS technical framework, we identified the following basis components:

- Operating System
- Web Server
- FTP Server
- Database server
- E-Mail infrastructure
- Statistics tool
- Links checker
- Scripting language

A preliminary question related to the POSS feasibility is whether or not to select Open source Tools only as technical framework components. We did so in the following section, although it was not technically required: sharing public sector specific software thanks to open source licenses could as well be done using proprietary server and applications. However, because mature OSS tools are available and to reinforce the credibility of the POSS feasibility study, the technical framework hereafter is based on OSS, although we do not consider this requirement as a technical one.

Operating System

Several open source operating systems are available.
Two alternatives are proposed following proven solutions on similar sites:

- Linux
- FreeBSD

Linux seems to be the more appropriate because of its worldwide success that guarantees support, documentation and updating.

Name: *Linux*.
Site: http://www.linux.org/
License: GPL

Linux is a Unix-Like kernel operating system that may be used for a wide variety of purposes including networking, software development, and as an end-user platform.

Name: *FreeBSD*.
Site: http://www.freebsd.org
License: GPL

FreeBSD is an advanced operating system derived from BSD UNIX. It offers advanced networking, performance, security and compatibility.

Web Server

A Web server is needed to connect to the Internet and exchange data. Apache HTTP is the most widely used throughout the Internet and has thus proved its reliability. It is therefore the proposed web server.

Name: *Apache HTTP*.
Site: http://httpd.apache.org/
License: Apache Software license

Apache has been the most popular web server on the Internet since April 1996. The March 2002 Netcraft Web Server Survey found that 54% of the web sites on the Internet are using Apache, thus making it more widely used than all other web servers combined.

Additional component

A security component must be added to provide cryptography services for the Apache web server.

Name: *ModSSL*.
Site: http://www.modssl.org/
License: BSD-style license

ModSSL provides strong cryptography for the Apache web server via the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1) protocols by the help of the Open source SSL/TLS toolkit OpenSSL[70], which is based on SSLeay[71] from Eric A. Young and Tim J. Hudson.

File Transfer Protocol (FTP) server

File Transfer Protocol (FTP), a standard Internet protocol, is the simplest way to exchange files between computers on the Internet. ProFTPD is an open source FTP server used by a lot of similar sites.

Name: *ProFTPD*.
Site: http://www.proftpd.net
License: GPL

ProFTPD grew out of the desire to have a secure and configurable FTP server. ProFTPD is not a hack based on any other server; it's an independent source tree from the ground up. A number of well-known and high traffic sites[72] use ProFTPD.

Database

The database will be intensively used to store information on users, applications, links, news and library information. MySQL is the world's most popular Open source Database.

[70] http://www.openssl.org/
[71] Documentation can be found at http://www.columbia.edu/~ariel/ssleay/
[72] See http://proftpd.linux.co.uk/sites.html

Name: *MySQL*.
Site: http://www.mysql.com/
License: GPL

MySQL is designed for speed, power and precision in mission critical, heavy load use.
PHP (the chosen server-side scripting language) allows accessing MySQL database easily through its MySQL support component (support for connecting to MySQL is built into the language).

Additional component

A component is added to handle the administration of MySQL over the web.

Name: *phpMyAdmin*.
Site: http://phpwizard.net/projects/phpMyAdmin/
License: GPL

Currently, phpMyAdmin allows to:

- Create and drop databases
- Create, copy, drop and alter tables
- Delete, edit and add fields
- Execute any SQL-statement, even batch-queries
- Manage keys on fields
- Load text files into tables
- Create and read dumps of tables
- Export data to CSV values
- Administer multiple servers and single databases

Possible alternative for database: PostgreSQL[73]

E-Mail infrastructure

The E-Mail infrastructure must include:

- ***A Message Transfer Agent (MTA)***. The MTA sends, receives, and delivers mails between servers within the Exchange System. The MTA also uses addressing and routing information to act as a relay, accepting messages from other servers and forwarding them to their destination.
- ***A mailing list manager.***

Message Transfer Agent

For the MTA, two alternatives are proposed:

- Exim

[73] http://www.postgresql.org

- Sendmail

They are equally used in similar sites.

Name: *Sendmail*.
Site: http://www.sendmail.org/
License: GPL

The Sendmail programme is a very widely used Mail Transport Agent. Administrators of similar sites we have contacted (SourceForge, Spirit, BerliOS) all use Sendmail as MTA.

Name: *Exim*.
Site: http://www.exim.org/
License: GPL

Exim is a Message Transfer Agent (MTA) freely available under the terms of the GNU General Public License.

Mailing List Manager

GNU Mailman is the most used mailing list manager on similar sites.

Name: *GNU Mailman*.
Site: http://www.list.org/
License: GPL

Mailman is a software to help manage electronic mail discussion lists. Mailman gives each mailing list a unique web page and allows users to subscribe, unsubscribe, and change their account options over the web. Mailman is fully compatible with both Exim and Sendmail MTA.
GNU Mailman requires the use of the Python interpreter (see below)

Additional component

Python is an *interpreted, interactive, object-oriented* programming language. GNU Mailman needs a Python interpreter to work.

Name: *Python*.
Site: http://www.python.org
License: GPL

MS Exchange Mail Takeover Tool

If there is a need for MS Exchange mail takeover, FetchMail can be used.

Name: *FetchMail*.
Site: http://www.tuxedo.org/~esr/fetchmail/
License: GPL

FetchMail retrieves mail from remote mail servers and forwards it via SMTP.

Statistics

As described in chapter 3, statistics will be gathered on the POSS site (page views, visits…). The Webalizer is widely used for this purpose.

Name: *The Webalizer*.
Site: http://www.webalizer.com
License: GPL

The Webalizer is a fast, free web server log file analysis programme. It produces highly detailed, easily configurable usage reports in HTML format, for viewing with a standard web browser.
It is used by thousands of systems around the globe (mozilla.org[74], linuxHelp.org[75]).

Link Checker

The link checker will be very useful to check the validity of all the links referred to from the site pages but also the links within the links database. It produces reports about all the links on the site after having checked their validity.
Linklint is an Open source Perl programme that checks links.

Name: *Linklint*
Site: http://www.mindspring.com/~bowlin/linklint/
License: GPL.

Scripting Language

The scripting language will be used to add dynamic contents into the HTML pages (different language contents, lists from the database…).

Name: *PHP4* (with MySQL support).
Site: http://www.php.net/
License: GPL

PHP (recursive acronym for PHP: Hypertext Preprocessor) is an open-source server-side scripting language for creating dynamic Web pages.

PHP offers a simple and universal solution for easy-to-programme dynamic Web pages. The intuitive interface allows programmers to embed PHP commands right in the HTML page.
Because of its wide distribution to a large community of users, PHP is very well supported. As an open source product, PHP enjoys the support of a large group of open-source developers. This community gives an excellent technical support to users, and bugs are found and repaired quickly. The code is continuously updated with improvements and language extensions to expand PHP's capabilities.

[74] http://www.mozilla.org/
[75] http://linuxhelp.dyndns.org/

Unlike other scripting languages for Web page development, PHP offers excellent connectivity to most of the common databases (including Oracle, Sybase, **MySQL**, ODBC and many others).

PHP is the natural choice for developers on Linux machines running Apache server software. PHP also supports HTTP sessions, Java connectivity, regular expressions, LDAP, SNMP, IMAP, COM (under windows) protocols. It also supports WDDX complex data exchange between virtually all Web programming languages.

PHP is today's fastest-growing technology for dynamic Web pages. According to the Netcraft survey[76] on the technologies actually in use on the Web, PHP can be found on more that 5 million domains, and is growing at a rate of up to 15% each month. PHP is available on over 36% of Apache Web servers – the most common server on the Web.

Table 1 summarizes the proposed tools.

Purpose	Tool(s)
Operating System	Linux or FreeBSD
Web Server	Apache HTTP, modSSL
FTP Server	ProFTPD
Database	Database: MySQL Database Manager (through the web): phpMyAdmin
E-Mail infrastructure	MTA: Sendmail or Exim List manager: GNU Mailman (Python needed)
Statistics	The Webalizer
Link Checker	Linklint
Scripting language	PHP4

Table 1 Basic Tools

Specific tools

In addition to the above-mentioned tools, specific tools will be used to manage the POSS service functionalities, as:

- Pooling software
- Retrieval and indexing tools
- Forum
- Mailing lists for newsletter

[76] http://www.netcraft.com/

- Security and POSS administration

The products selected are out-of-the-shelf or adaptable components.

Pooling software tool

SourceForge provides a framework for cooperative developments and pooling software, including software indexing, mailing lists, forums and security implementation. The versioning tool used by SourceForge is CVS (Concurrent Versions System) that is the open standard for version control. Security components can be coupled to CVS to allow x509 certificate to be used for the key exchange phase and encryption.
Secure Shell (SSH), sometimes known as Secure Socket Shell, is a UNIX-based command interface and protocol for securely getting access to a remote computer. It can also be used to secure file transfers with certificates. Both ends of the client/server connection are authenticated using a digital certificate.

Name: SourceForge[77].
Site: http://sourceforge.net/projects/alexandria-dev
License: GPL

SourceForge is widely used by similar sites (SourceForge.net[78], BerliOS[79], Spirit[80]).
SourceForge provides the following functionalities:

- Building and managing software database (MySQL)
- Customisable software index
- Software browsing by predefined fields.
- User management for software area (permission to manage software files, to do modification to software…)
- Mailing lists
- Discussion forum

Name: Concurrent Versions System (CVS).
Site: http://www.cvshome.org/
License: GPL

CVS is the Concurrent Versions System, the dominant open-source network-transparent version control system.

Name: OpenSSH
Site: http://www.openssh.com/
License: BSD

[77] SourceForge 2.7 open source will be released as "Alexandria 2.7" during the month of August 2002.
[78] http://sourceforge.net
[79] http://www.berlios.de/
[80] http://www.euspirit.org/

OpenSSH is a free version of the SSH protocol suite of network connectivity tools that increasing numbers of people on the Internet rely on. OpenSSH encrypts all traffic (including passwords) to effectively eliminate eavesdropping, connection hijacking, and other network-level attacks. Additionally, OpenSSH provides a lot of secure tunnelling capabilities, as well as a variety of authentication methods.

Contractual engagement process

In chapter 2, we described the contracting process.
A software component must be developed in order to monitor the user certified identification, the validity of his agreement on the license terms and the additional attribution of jurisdiction.

Software size evaluation

One of the entries in the software index is the size of the software. This size can be calculated by an automated tool.

Name: *SLOCCount*.
Site: http://www.dwheeler.com/sloccount/
License: GPL

SLOCCount, developed by David Wheeler, proposes a set of tools allowing extraction of the size of a software directly from the source code package. The tools are able to automatically extract not only the number of lines of source code (avoiding code comments) but also to extract important parameters like COCOMO[81] estimates for re-development cost and time.

Text search engine

Search engine will be very useful to provide search functionalities across the entire site.
Jakarta Lucene is a full-featured text search engine written entirely in Java. It is a technology suitable for nearly any application that requires full-text search, especially cross-platform.
Jakarta Lucene is an open source project available for free download from Apache Jakarta.

Name: *Lucene*.
Site: http://jakarta.apache.org/lucene
License: Apache Software License

Links, news and library databases

The MySQL Functions of PHP[82] allow easy access to MySQL database servers.

[81] The COCOMO cost estimation model is used by thousands of software project managers, and is based on a study of hundreds of software projects.

The databases can then be interfaced (insertion, deletion, modification and search) through the web thanks to PHP embedded in HTML pages.

Administration and security

Administration services can also be implemented thanks to PHP embedded in HTML interface pages.
The security layer for accessing the POSS portal site must also be developed. Hints to develop it using both built-in Apache authentication and custom PHP code can be found at
http://www.devshed.com/Server_Side/PHP/UserAuth/page1.html

Table 2 gives a summary of the specific tools.

Purpose	Tool(s)
Software pooling	SourceForge, CVS, OpenSSH
Software Size extraction	SLOCCount
Full text indexer	Lucene
Contracts acceptance	To develop
Databases (links, news and library)	To develop
Administration and security	To develop

Table 2 Specific tools

Figure 24 shows the various components of the POSS architecture

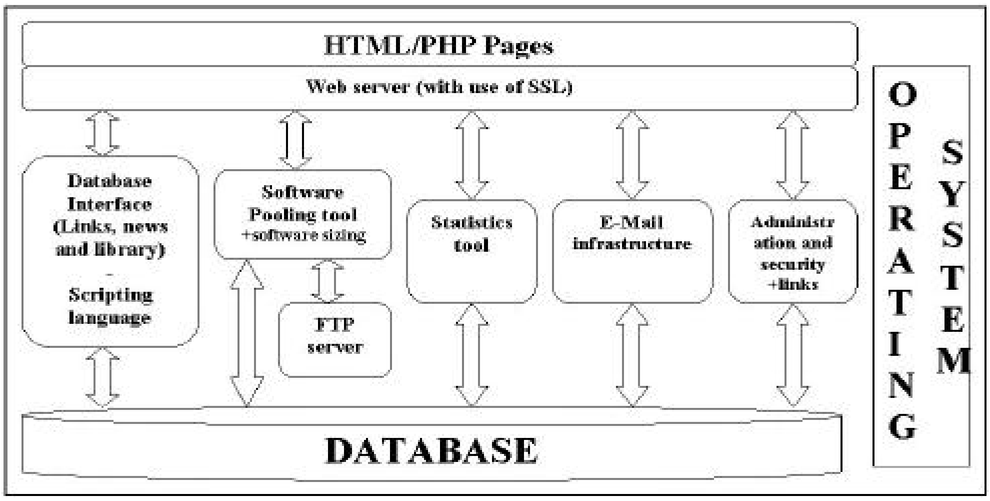

HTML/PHP Pages
Web server (with use of SSL)
Database Interface (Links, news and library) - Scripting language
Software Pooling tool +software sizing
FTP server
Statistics tool
E-Mail infrastructure
Administration and security +links
DATABASE
OPERATING SYSTEM

Figure 24 POSS architecture

[82] See http://www.php.net/manual/ro/ref.mysql.php

Set up of the POSS

This section details the different tasks that will have to be fulfilled in order to set up the POSS portal site.

Detailed functional analysis

Definition

Based on the functionalities described in chapter 3, a detailed overview of the system and its desired functionalities must be provided.
The specifications and requirements documents must be prepared.
The European Commission must approve those documents.

Responsibilities

POSS integrator

Deliverables

Specifications and requirements documents
Software form
List of Software index fields
Link form
List of link index fields
News form
List of news index fields
Library material form
List of library index fields
User registration forms

Detailed technical design

Definition

The objective of the detailed technical design is to produce an accurate description of the system. The components (hardware and software) and the interactions between them must be clearly defined.

Responsibilities

POSS integrator

Deliverables

Technical design documents

Standards

Definition

The technical framework of standards must be defined for the coding of all the software that will be registered on the POSS (naming conventions, packaging convention…).
The documentation policy must also be defined, stating a.o. the minimum documentation that is required for each software.
Security and backup policy are defined.

Responsibilities

POSS IT Experts

Deliverables

Coding standards
Documentation policy
Security policy
Backup policy

Contracts

Definition

The various contracts (see chapter 2) must be defined:

- The contract between the provider/author and the POSS. It is a kind of "*power of attorney*" to distribute the software
- The contract between the POSS and the users who want to download software from the POSS

Responsibilities

POSS Law Experts

Deliverables

Contract between licensor/provider and POSS
Download contract between POSS and users

Installation and configuration

Definition

Installation and configuration of the operating system (with the Python package) and the tools defined in the previous paragraph.

- Web server
- FTP server

- Database tools
- E-Mail infrastructure
- Statistics
- Link checker
- Scripting language
- Software sharing tool
- Software sizing tool
- Text indexer

Responsibilities

POSS integrator

Deliverables

Installation procedures documents

Links database creation and interfacing

Definition

The links database must be created and interfaced through HTML pages with PHP scripting.

Responsibilities

POSS integrator

Deliverables

Links database
Interfacing HTML pages

News database creation and interfacing

Definition

The news database must be created and interfaced through HTML pages with PHP scripting.

Responsibilities

POSS integrator

Deliverables

News Database
Interfacing HTML pages

Library database creation and interfacing

Definition

The library database must be created and interfaced through HTML pages with PHP scripting.

Responsibilities

POSS integrator

Deliverables

News Database
Interfacing HTML pages

Security layer development

Definition

SourceForge provides security feature for the software area. However, a security layer must be implemented to control access to the POSS. Particularly for the part related to the acceptation of the contracts.

Responsibilities

POSS integrator

Deliverables

Security layer

Existing illustrations of such possible architecture

Berlios

Site: http://www.berliOS.de

The main goal of BerliOS is to support the different interest groups in the area of open source software (OSS) and thereby to offer a neutral mediator function. The target groups of BerliOS are on one hand the developers and users of open source software and on the other hand commercial manufacturers of OSS operating systems and applications as well as support companies.

The following information and services are offered to*users*:

- ***Documentation***
  The worldwide emerging documentation about open source software is classified by BerliOS and is provided to the public via (DocsWell) the worldwide web. The documentation contains HOWTO and FAQ documents, free books and books directories, as well as standards and other documents. Besides the offering of English documents a main goal of BerliOS is the provision of information in German language.
- ***Databases***
  Users can consult extensively about Open source distributions and application software (SourceWell), hardware and software products as well as support services and companies (SourceBiz).
- ***News & Events***
  News and event notes (BerliOS News) about Open source topics are proposed in a compact form.
- ***Solutions for particular trades***
  Existing solutions in companies are introduced and documented (SourceLines). The experience gained during realization and deployment can then be reused in other companies. BerliOS provides a technical infrastructure for the exchange of experience between users and a mediator function between users and developers as well as support companies.
- ***Software Exchange***
- ***Search Engine***
  Besides search engine enabling users to find information on the BerliOS Web server, an additional directory of other worldwide existing search engines is offered.

A server-based infrastructure (BerliOS Developer) is provided for the developers of Open source software. The administrative tools support the joint coordinated software development between people who live and work at different geographical places.

Software behind the BerliOS site are:

- HTTP server: Apache Web server with ModSSL
- Database: MySQL database server and PhpMyAdmin
- Mail: GNU Mailman
- PHP
- SourceForge

Spirit

Site: http://www.euspirit.org

SPIRIT is a pioneering project partially funded by the European Commission's Fifth Framework Programme. It accelerates the uptake of software and other resources to facilitate the implementation of economically viable and effective regional health care solutions.

SPIRIT provides freely available resources that will enable better citizen-centred care in Europe and around the world.

The project identifies and classifies best practice open source software applications and components from both existing ongoing projects, and planned projects. Sources for software include government agencies, medical teaching institutes, and other health care providers.
SPIRIT assembles a community of dedicated professionals with similar goals, and provides a common meeting place that will increase the available base of open source software for health care. SPIRIT services include disseminating open source research results, groupware applications, audio/video conferencing facilities, mailing lists, and web site hosting for open source health care projects.
Software collected by the project are distributed via CD-ROM and the SPIRIT web portal. SPIRIT is an exciting opportunity for everyone involved in health care informatics. For the open source health care community, SPIRIT offers a way to solicit involvement and increase awareness on open source health care projects.

Software behind the Spirit site are:

- SourceForge 2.0
- HTTP server: Apache 1.3.20
- FTP server: Proftpd 1.2.4
- Database: PostgreSQL 7.2
- MAT: sendmail 8.12.1
- PHP 4.02

Atica

Site: http://www.atica.pm.gouv.fr/

The main objectives of the Agency for Information and Communication Technologies into Administrations are:

- the definition of a general framework of interoperability between administrative information systems
- the recruitment of computer scientists
- the regular diffusion of information both to the administrations and to the public.

Software behind the Atica site[83] are :

- Operating System: FreeBSD version 3.4
- HTTP server: Apache 1.3.6
- Database: MySQL 3.22.32 and PhpMyAdmin v2.0.5
- MAT: Sendmail 8.8.8
- PHP 3.0.7

[83] http://www.atica.pm.gouv.fr/bouquet-libre/tests/config_freebsd.shtml

5. Maintenance and Interaction

Objectives

The objective of this chapter is to identify the different roles of the POSS portal site and the responsibility of each role in the maintenance processes.

The first part describes the general principles of a cooperative service management. The site must be administrable from several locations in Europe.

In the second part the different roles of the POSS are briefly described with an overview of the responsibilities.

Afterwards, the maintenance processes are described. They are divided in three categories:

Components maintenance
Contents maintenance
Services maintenance

The components maintenance includes only hardware and basic software (web server, forum tool, e-mail infrastructure) maintenance.
The contents maintenance includes maintenance of software index and home pages, links, news, and library indices.
The services maintenance includes members registrations, legal framework, forums, mailing list, surveys, and statistics.
For each maintenance process, the responsibilities are identified.

Then, the problem of the data integrity is described together with the solution proposed for the POSS site.

Finally, the possible contribution of organizations in Europe is evaluated.

Principles of a cooperative service management

The POSS is a European portal site. The administration and the monitoring must be done in a cooperative way and be as decentralized as possible.

Even though a specific organisation may be recognised as being responsible for the POSS site, several national agencies (Atica, BerliOS…) should be involved in the POSS administration and monitoring. This should ensure to reach a maximum number of people in several countries. This is very important to avoid limiting the POSS services to leading countries and to ensure multilingual features of the POSS portal site.

Moreover, low-level responsibilities must be delegated to a "OSS group" responsible (software responsible). It means that the software responsible receives rights to attributes roles to developers, manage mailing list and forums inside his group.

Those constraints imply a web-based management. It means that most of the administration could be done through the web with a secured user authentication.
The user interface must then be clear, easy to use and multilingual.

The service roles

Different groups of users will access the POSS Portal site. Each of them has a defined ***role***, with precise ***responsibilities***.

System administrator

The system administrator is the *Super User* of the site. He has all the access rights in the POSS and is able to give or to prohibit access to the site and its services. He will be in charge of the management and operations of the POSS site.

This role includes:

- Hardware configuration management: servers, network…
- Software configuration management: several software compose the system (application sharing, forums, e-mail infrastructure, web server, ftp server…) and the system administrator must maintain them.
- Maintenance of the software developed for users management, forums management, mailing lists management and security.
- Maintenance of the statistics tool.
- Security maintenance.

Evaluator

The evaluator will evaluate the software that is submitted for sharing on the pooling site. His evaluation criteria will check two aspects: "*opportunity*" and "*quality*" (see below, software submission). The evaluator is the warrant of the quality of the software available on the POSS: he must be an expert in his job (developers, architects, linguists).
This role must be reserved for IT experts aware of the needs of the European administrations.

The evaluators have the responsibility to accept or reject the submitted software.

Controller

The controllers form the "*steering committee*" of the POSS portal site. They are responsible for the quality of the site contents (except for software that is under the responsibility of evaluators) and for the users information.
This role implies:

- Users management (create, delete)
- Forums management (create, delete)
- Mailing lists management (create, delete)
- Management of links, news and library database
- General public forums moderation
- Newsletters and surveys preparation

The library materials must be submitted and accepted before being stored in the database but links and news can be directly added and must then be checked.
Rather than an expensive permanent control, regular "*check points*" (e.g. weekly) must be set up.

Controllers have also the responsibility of appointing evaluator and registering software responsible (see below).

Software responsible

The software responsible is the coordinator of a development project. He is the single contact point for software. He will be in charge of transmitting the project information and its status on the POSS Portal site and to administrate his software area.
This includes:

- Software developers management (add, delete and edit)
- Moderation of forums related to his software (after the creation done by the controllers)
- Management of mailing lists related to his software (after the creation done by the controllers)
- Software homepage management

They are registered together with their software. Software must always have an active software responsible.
This role implies the acceptance of the POSS chart.

Developer

A developer can also submit a project. But his role consists mainly of developing the software and providing support and maintenance for it. They can be contacted directly by the users. The registration of several developers is interesting to get support quickly.

User

A user registered on the POSS has no role in the development of the OSS. He is a potential user of the software and certainly an adept of open source software. He can add links and news and submit library materials.

Maintenance process

Maintenance is an ongoing and iterative activity that takes place after the system has gone into production.
Figure 6 shows the various steps of the ITIL[84] iterative process of maintenance.

[84]IT Infrastructure Library (ITIL) is a consistent and comprehensive documentation of best practice for IT Service management. (http://www.itil.co.uk/index.html)

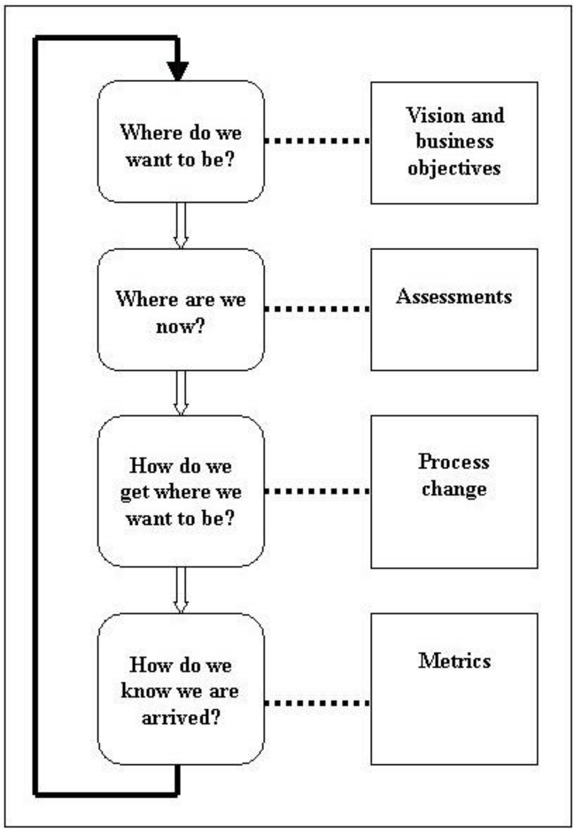

Where do we want to be?
Vision and business objectives
Where are we now?
Assessments
How do we get where we want to be?
Process change
How do we know we are arrived?
Metrics

Figure 6 Maintenance process

The first step of the process is represented by the question "*Where do we want to be*?". The answer describes the objectives to which the system should evolve. Several hints can help to define this objective, for example:

- Problems submitted by users to be corrected
- New functionalities proposed by users
- New release of system components (web server, database…)
- New legislation (software patent, license…)

The second step, represented by the question "*Where are we now?*", is the assessment of the current situation.

The third step includes the evaluation of the effort to be made to reach the objective from the current situation. New processes are then defined and initiated.

The fourth step is the measurement of the objective achievement and determination of the new initial state for the next cycle.

Maintenance can be:

- ***Correction***. The process includes the diagnosis and correction of errors not discovered during design, implementation, or testing.
- ***Adaptation***. The process includes the modification of the system to properly interface with technology changes.

- ***Improvement***. The process includes recommendations for new capabilities, modification to existing functions, and general enhancements received from users.

The POSS site must set up a maintenance plan that defines the goals and constraints for maintaining the site over a period of time.
The maintenance is divided into three types:

- The components maintenance
- The contents maintenance
- The services maintenance

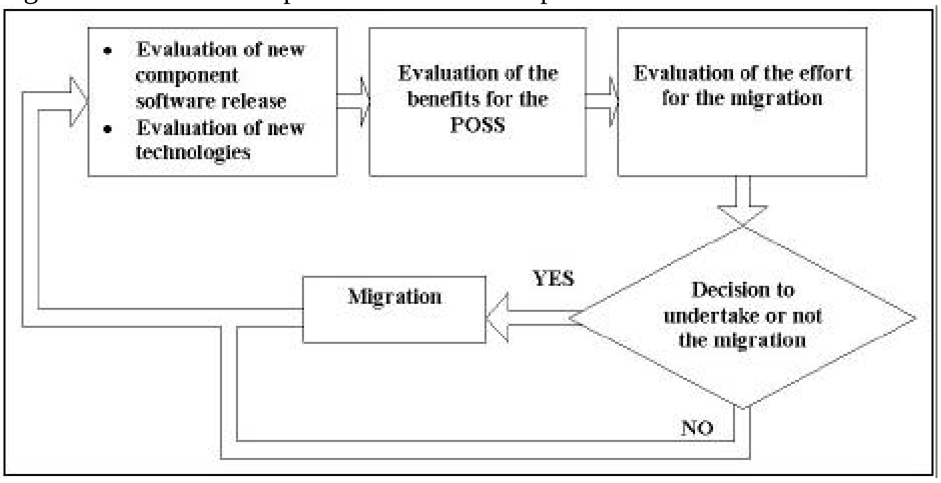
Components maintenance

The components maintenance is an "adaptation" maintenance.
On a regular basis (e.g. yearly), the system components must be evaluated against new available technologies.
Most of the POSS components should be open source and open source evolves very quickly (new releases, patches…). The regular re-evaluation is then very important.

The ***system administrators*** will be in charge of

- Evaluation of new component software release and new technologies available.
- Evaluation of the benefit for the POSS
- Evaluation of the effort for the migration
- Decision to undertake (or not) the migration
- Migration

Figure 7 shows the components maintenance process.



• Evaluation of new component software release
• Evaluation of new technologies
Evaluation of the benefits for the POSS
Evaluation of the effort for the migration
Decision to undertake or not the migration
YES
Migration
NO

Figure 7 Components maintenance process

Contents maintenance

Software

The “software maintenance” includes

- Addition of new software in the index
- Modification of software
- Deletion of software
- Versioning management

Software addition

When a member submits software, it is evaluated for acceptance by the***POSS evaluator***. It must be evaluated in terms of:

- *Opportunity*: to determine if the software is interesting for other administrations and if there is a need for such software.
- *Quality*: code and documentation quality must be evaluated.
- The code must respect standards, include sufficient comments and be easily adaptable (configuration files, no hard-coded parameters…).
- The documentation must include at least the technical description, a user manual and an administration manual.

The ***evaluator*** conclusions are then transmitted to the***POSS controller***. According to the conclusions, the controllers register the software on the POSS or warn the member that the software does not answer the POSS requirements.
The registration includes the addition of an entry in the software index and the upload of software files (HTML presentation pages, binary distribution, sources and documentation) if the provider doesn’t have server facilities. Space must be allocated on the servers (FTP server for software files and web server for software home page).

If the software is registered on the site, a***software responsible*** is also registered (usually the submitter himself) and receives the rights to administrate the software area (add developers, manage forums and mailing list related to the software…). A forum and a mailing list dedicated to the software are also created.

If the software is rejected, it can be improved by the submitter and re-submitted later.

Figure 8 shows the software addition process.

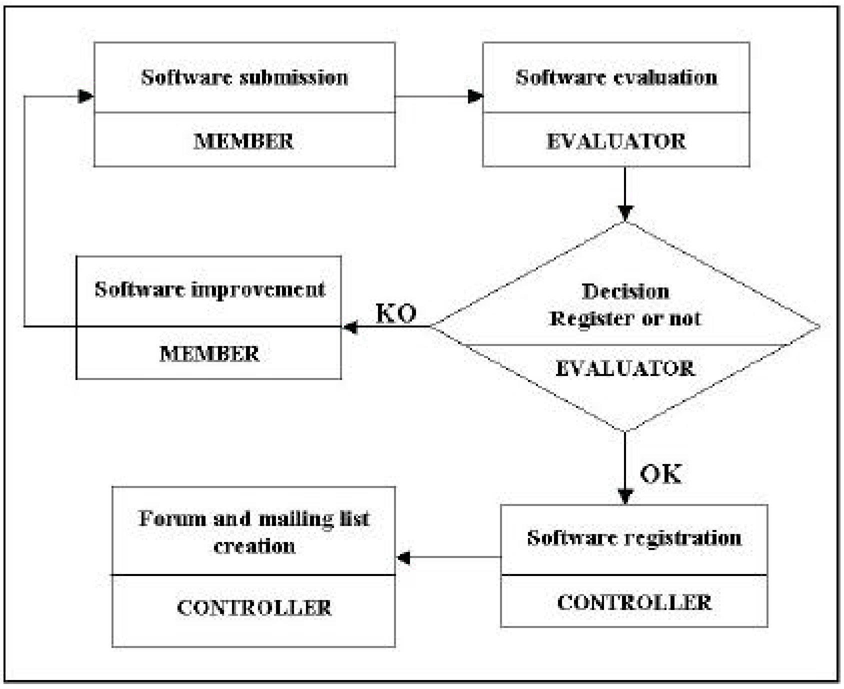


Figure 8 Software submission process

Software Modification

Each software modification (new release, updated description, new platform supported…) must be evaluated (***evaluators***). If the evaluation is positive, the modifications have to be reflected in the index (***controllers***) and possibly, new software files have to be uploaded. If the provider hosts the software, coherence between information contained in the index and the unloadable files (version number) must be under the responsibility of the***software responsible***. To ensure regular follow up of the software evolution, feedback from the ***software responsible*** could be regularly required.

Software deletion

Controllers have the rights to delete software from the index.

Versioning management

The versioning management will be done by the CVS tool described in the chapter 4. When a new version of software is released, the previous versions remain available with a different version number. The changes between the new and the old version will be entered in the versioning tool and available for users.

Table 3 shows the responsibilities for software maintenance.

Software registration and facilities (forum and mailing list) set up	Controllers
Software evaluation	evaluator(s) reserve the right to determine the appropriateness of any entry in the software index
Software edition/deletion	Controllers

Software registration and facilities (forum and mailing list) set up	Controllers
Software indexing	POSS evaluators, administrators and controllers

Table 3 Software maintenance responsibilities

Links

Members can add links dynamically. The links are immediately added in the links database and can be consulted by other users.
However, the ***POSS controller*** must check the links pertinence regularly.
Moreover, tests must be done regularly to detect broken links in the database. This can be done by an automated tool (link checker). The link checker set-up and configuration is under the responsibility of the ***POSS system administrator***.
Besides the links contained in the database, all the links on the POSS site pages must also be checked.

Table 4 shows the responsibilities in links maintenance

Check interest of link	POSS controllers
Maintenance of the link checker	POSS administrators
Maintenance of the index	POSS controllers

Table 4 Links maintenance responsibilities

News

Members can add news dynamically. To avoid publishing delay, the news is immediately added in the news database and can be consulted by other users.
However, the ***POSS controller*** must check the news pertinence regularly.

Library

Members can submit library material. The ***POSS controller*** must check its pertinence and its quality before adding it in the library database.

Figure 9 shows the library material submission process.

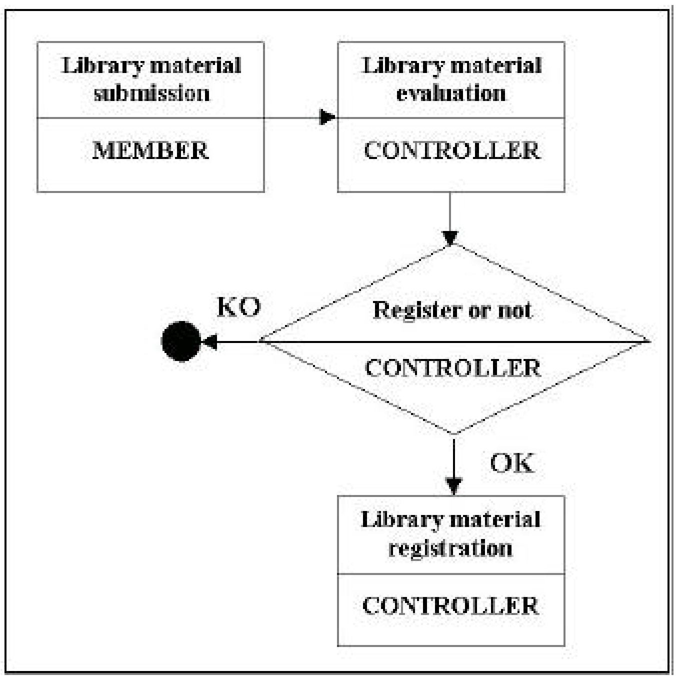

Library material submission
MEMBER
Library material evaluation
CONTROLLER
KO
Register or not
CONTROLLER
OK
Library material registration
CONTROLLER

Figure 9 Library material submission

Services maintenance

Member management

Member's management will determine who will access specific areas of the system (user identification) and what privileges a person will get.
Member's management aims at protecting the site from attacks while offering an adequate service to each member profile.

Member's management includes also the registration on the POSS portal site. The member's registration is done on the web thanks to forms that are submitted and accepted/rejected by

- ***Controllers*** for *evaluators, controllers* and *software responsible* and *users*. A core group of controllers will be formed at the POSS launching.
- ***Software responsible*** for *developers*

After acceptance, the member receives his login and password.

Table 5 shows the creation procedure for the different profiles.

Profile	Creation procedure	Responsibility of
System administrators	One administrator will exist from the start. System administrators can only be created/deleted/edited by another system administrator	Administrators

Profile	Creation procedure	Responsibility of
Software responsible	Software responsible is responsible for software development. If the project is small and implies only one developer, he cumulates the roles of software responsible and developer. The software responsible profile is created by the administrators or the controllers upon receipt of the software data.	Administrators Controllers
Users	A registration form will be available on the site; an automatic e-mail is sent to the address submitted by the user, the user is created on receipt of his answer to this e-mail.	Administrators Controllers
Developers	When the software is submitted, an initial list of developers is registered by the controllers. After the software registration, the software responsible can add/suppress developers for his project.	Controllers Software responsible
Controllers	The identified controllers will be registered by the administrators from the start of the POSS. Registration form will be available on the POSS site to apply for this role. A controller can only be created/deleted/edited by the system administrator or another controller.	Administrators Controllers
Evaluators	This role is very important since evaluators are the warrant of the quality of the site. This role must be restricted to experts that are registered by the controllers. An evaluator registration form will be available on the POSS site to apply for this role.	Controllers

Table 5 Creation procedure

Legal framework

Laws concerning computer software have to evolve quickly to keep pace with the quick evolution of computer technology. This fact must be taken into consideration for the POSS. A way to keep in touch with the open source legislation (software patents, e-commerce…) must be defined. This could be a contract with ***lawyer consultants***.

Security maintenance

The objectives of the Security Maintenance are as follows:

- Ensure the integrity of the infrastructure, of the applications and of the information.
- Ensure a properly secure operational environment for the POSS infrastructure for physical, network access and access control levels.
- Ensure the ability to be audited through the maintenance of reference information (documentation, audit trails, logbook).

The security maintenance is under the responsibility of the **system administrators**.

Forum

The ***system administrators*** are responsible for the maintenance of the software providing the forum functionalities.

A form must be available to request a forum creation.
The ***controllers*** are responsible for creation and deletion of forums on request.

The forums moderation is under the responsibility of:

- The ***Controller*** for the general and public forums
- The ***Software responsible*** for specific forum related to software under his responsibility.

Deletion policy can also be set up by the ***controllers*** to delete forums that have not been visited during a certain period of time.

Mailing lists

A mailing list is created automatically for software registered and the software responsible receives the right to manage this mailing list.
Members can also ask for the creation of a specific mailing list. Request must be sent to ***controllers*** who create the new mailing list and give the member the rights to manage it.

The ***controllers*** also manage the general mailing list.

Table 6 shows the mailing lists maintenance responsibilities.

General mailing lists	POSS controllers
Software related mailing lists	POSS software responsible

Table 6 Mailing lists maintenance responsibilities

Newsletter

Newsletters should be sent regularly (e.g. monthly) to members. They must be prepared and sent by the ***controllers***.

Survey

The key objective of this component is to provide valuable information for the following benchmarks[85] :

- Design of the web site for common community scenarios - how well does the site support the tasks the visitor is trying to accomplish
- Community growth - how well do the site services, Internet indexing and dissemination events attract new community members.
- Personalisation suitability - how well does the site address the personalisation requirements of visitors and members
- Technical performance – how available and responsive is the POSS infrastructure.

Survey must be regularly prepared to understand the users needs and improve the functionalities of the POSS.
The ***controllers*** are responsible for

- Preparing surveys
- Sending surveys
- Collecting results of surveys
- Writing results reports

The evaluation of the users' needs and the new functionalities proposed by users is under the responsibility of the ***controllers***. They must also take the decision of implementing (or not) new functionalities in consultation with the ***system administrators***.

Statistics

Statistics will be gathered and published by an automated tool. The tool set-up and configuration is under the responsibility of the ***system administrator***.

Backup

Backups are organised in such a way that a copy of the last versions of the systems software, configuration files and data files is always available in a safe place. Previous releases of software are always backed up before installing a new version. Recovery procedures have been tested and their duration is known.

Backup management aims at restoring the system as soon as possible in case of hardware, software or security failure leading to software or data corruption or even loss.

The ***system administrators*** will operate the system

[85] Customers.com ® Quality of Experience Benchmark, Patricia Seybold Group's Customers.com ® Strategic Planning Service,August 31, 2000

The data integrity

First, the data integrity will be managed by avoiding maintenance of multiple sources. The different indices will link to status information, rather than contain it.
The information will be maintained on the software home page located by the software provider or on the POSS server.

However, it is not enough. The information contained in the index must always be exactly the same as the information contained on the software home page. For example, if the version is said to be 2.01 in the index, the software home page must presents the version 2.01 as the last version available. And the downloadable version must be 2.01.

This integrity is expensive to maintain because it requires a lot of detailed checks. The solution proposed for the POSS is an agreement between the ***software responsible*** and the ***POSS controllers*** to ensure the data integrity. The ***software responsible*** must be the warrant of the data integrity between the software index and the software home page.

Possible contributing organisations

The first organisations concerned are obviously the European administrations. They must play a key role in the POSS portal site (provide software, opinions, success stories…).

Other organisations like Inria[86], BerliOS[87], Linux Center[88], Spirit[89] set up similar services. These organisations could be contacted so that software that could be of interest for the administrators can be referenced to in the POSS index.

National organisations (like ATICA in France) can also play a role to make the POSS portal site better known and recognised by the administrations. They can also contribute to library, links and news indices.

Specific parts of the administration of the POSS portal site could also be proposed to national agencies.

[86] http://www.inria.fr
[87] http://www.berlios.de
[88] http://www.linux-center.org
[89] http://www.euspirit.org

6. POSS Costs and Financing

Objectives

In the hypothesis a Pooling Open source Software service should be constructed, the purpose of this section is the estimation of the costs.

Prototyping costs (analyse and development)

Deployment costs requiring not only technical but also considerable informational efforts and translations;

Maintenance and operation: the periodic re-evaluation, upgrades and POSS application life cycle.

In regard to the costs, we will investigate the financing options: public funding, software related fees (subscriptions), supporting organisations and governments, supporting funding programmes (EU / Nationals) and possible agreements with IT vendors/distributors or service providers.

Understanding the Cost and Business models

Globally the cost of any IT project can be divided into Direct and indirect costs

The direct costs are twofold: – Assets (including the labour of implementation) and Exploitation (Operation / administration-maintenance)

Assets includes:
– Hardware: servers, clients, peripherals, network

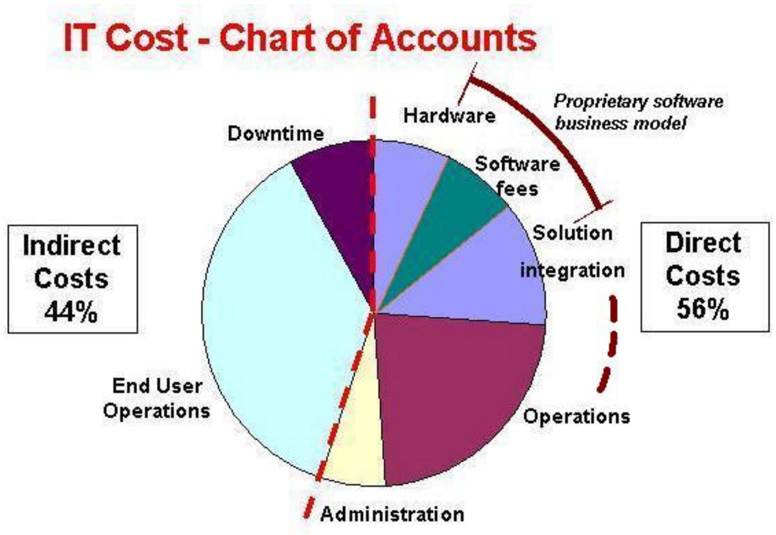


Figure 29 classical IT chart of accounts

- Software and solution integration are the budget parts where the use of open source components may have the strongest influence: In a proprietary business model, software fees cover operating systems, applications, utilities, web management tool etc. By purchasing all licenses from a single provider, the customer can expect a single contractor, a single documentation standard, a common look and feel and tested interoperability between licensed components (reducing then the solution integration budget). Similarly during the operation phase, a part of the software investment (from 12 to 18% usually) will be dedicated to maintenance fees.

If applying the open source business model, the part of the budget dedicated to license fee will be reduced, although still exiting (OSS components are not for free). At the contrary, the solution integration budget will be higher.

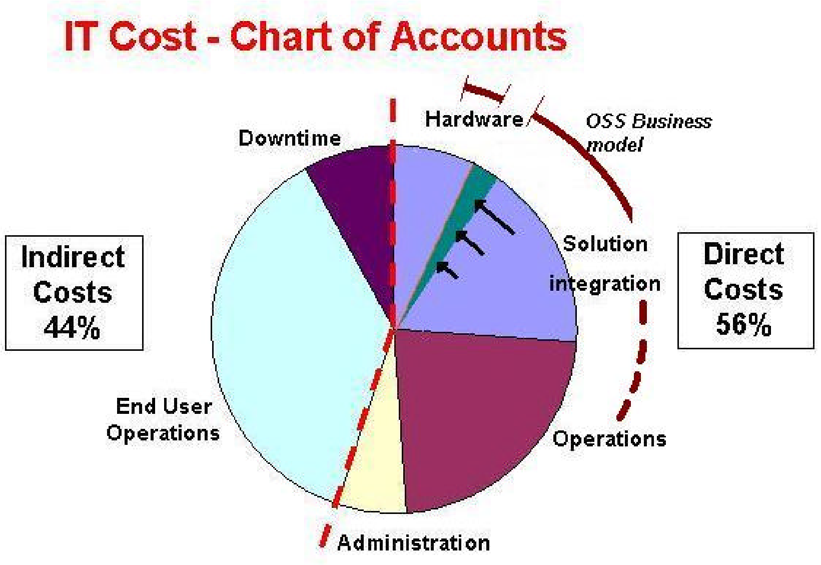


Figure 30: In the OSS business model, service providers compensate the lack of license fees by solution development and integration, support services during operations, resources from new markets as embedding OSS in hardware products. Users benefit from reduced license fees (mainly when a scaling effect may be obtained) but may face higher development costs and sometimes migration costs increasing the solution integration budget.

In general, the reason why the investment for solution integration may be globally as important or higher in the OSS business model may be due to migration costs from proprietary formats (not applicable in the POSS case) or the “vendor-less” effect due to the necessity to integrate multiple components (applicable in the POSS case, as the solution is specific and requires numerous components).
The integration effort may be largely compensated in large replicable projects (scaling effect) but this is not applicable in the POSS development itself:

- Preliminary benchmarking study in a fast moving “market”): feasibility, requirements, design, tool selection; this part may be increased with the POSS software due to its specific character and to the “vendor-less” effect: it requires to search for best practice and to check component integration.
- Demonstration to management,
- Prototyping, testing, pilot phase and implementation in production;
- Documentation of all aspects of the solution

In addition to the above deliverables, the various open source service companies or integrators can also deliver substantial parts of the subsequent operation tasks:

- Service Level Agreement regarding maintenance / support, help desk
- Service Level Agreement regarding the POSS content management
- Seminar cycles to inform public administration
- Service Level Agreement related to the participation to a scientific and evaluation committee
- Optional assistance related to legal issues and patent investigation

- Outsourcing of human and operational cost (operators, rooms, back-ups, security, network management, website management etc.)

Near to the operation is the “Administration”: Considering the POSS, it must cover:

- Budget/ finance and management/administration itself (in general, this part includes also HR costs as training for the IT team and for internal end-users, but this is not applicable to the POSS).
- Contractual framework management and legal advising concerning the POSS contractual structure evolution.
- Quality control and reporting to the POSS sponsors
-

Cost estimation

The POSS is not a simple web site (linked HTML pages where the cost is linked to the quality of presentation, the number of pages and the number of translations). The content part (topics and sub-topics) of the POSS includes such web pages, but most of all, the POSS is a portal open to a wide set of possible services and to national sites:

These services (described in chapter 3) include for example the user identification, the search engine, the newsletter expedition, the contracting for downloading software etc.

The services will be supported by back office databases: for pooled software, for links, forums, mailing list, registered users, list of news, list of services providers etc.

Each of these services and database is related to an application, for example: create, update, retrieve, annotate, delete and attach documents to a software notice.

The level of customisation, quality and automation of these applications is variable and will have a direct impact on the development costs. For example:

- A mechanism can issue warning mail to the software provider if he fails to update his project description every 6 months;
- The validation of the software description can be developed and automated;
- The presentation of links and news to the user can be prioritised according to his language;
- Etc.

The first estimation we provide hereafter is therefore to discuss according to ambitions (services to provide or to cut) and available budget, but in any case the POSS – now unknown – will require a complex interaction of various actors, that will require a serious investment, commitment and control, at least during the five first years[90].

For each work package, the effort is estimated in thousands euros.

The global estimation is the following:

Pilot portal prototype	510.000 to 1060.000
Deployment	340.000
Yearly costs	975.000

TCO on 5 years	6.000.000

[90] The yearly operational cost of the Canadian government service KES (Knowledge Exchange Service) is 800.000 can$ (One country, two working languages). The approach of this service, that plans to open its web portal at the end of year 2002, is quite similar to the POSS approach.
HTTP://knowledgexchange.pwgsc.gc.ca

Prototyping / Pilot phase

The Pilot is a prototype with full functionalities, but a limited number of languages (2) a limited content (number of pages), a limited number of links and records in all the POSS databases (software, library, users, providers etc.)

The work package / estimated budget are:

WP	Title	Deliverable	Cost (000)
01	Collection of a pilot panel of software contributors	Based on the result of questionnaires (Feasibility Study Appendix; + expression of interest following the feasibility study publication)	30
1	Detailed functional analysis	Specifications and requirements documents Software form List of Software index fields Link form List of link index fields News form List of news index fields Library material form List of library index fields List of Service provider index fields	30
2	Contractual framework implementation	POSS chart, Mandates Licenses Contractual conclusion business modelling and workflow	20
3	Detailed technical design	Technical design documents (hardware, software and interactions). Technical framework of standard	15
4	POSS Quality Standards definition	Coding /naming Classification Documentation policy Security Backups SLA for system administration, evaluations, QC	30

WP	Title	Deliverable	Cost (000)
5	POSS pilot development and integration	Installation, Integration and configuration of the operating system and all tools and functionalities. Web server FTP server Database tools E-Mail infrastructure Statistics Link checker Scripting language Software sharing tool Software sizing tool Tests, presentation, corrections	250 to 800
6	Links database creation and interfacing	Links database Interfacing HTML pages	30
7	News database creation and interfacing	Links database Interfacing HTML pages	30
8	Library database creation and interfacing	Library database Interfacing HTML pages	30
8	Service providers database creation and interfacing	Providers database Interfacing HTML pages	30
10	Security layer development	Roles and security matrix Security layer and authentication for: Administrator Content Contributor Licensor User Members Service providers	45

Deployment phase

After the prototype will have been tested, corrected and approved, the deployment phase is necessary to:

- Populate the databases and the POSS content as web site and web portal: pages, links, software,
- Translate the pilot phase services in several languages
- Implement a productive hardware infrastructure
- Set-up the POSS administration (conclude SLAs with the various operational actors)

WP	Title	Deliverable	Cost (000)
	Content enlargement	Topics / sub topics Main pages/ Software descriptions	40
	Translations	Depending on the number of languages (contractual framework and main pages)	100
	POSS operational technical infrastructure	Servers Security / backups Lines / routers /networks	100
	POSS operational organisational infrastructure	Administration System administration Evaluators Controllers Legal service Translation procedures Quality procedures and monitoring	100

Operation and Maintenance (yearly)

Even based on a decentralized cooperative framework of agreements delegating most of the POSS content administration to volunteers or to software providers, the POSS operation and maintenance must be based on a series of SLA's defining expectation from the various actors.

The role of software (project) responsible, developer and user are not included in the cost table below (they are volunteers), and will be generally defined into the POSS chart.

Globally the long-term maintenance of the service represents, by far, the most important budget.

WP	Title	Deliverable	Cost (000)
	System administration	To define by a SLA Hardware management and availability: server, router, network Maintain security Maintain software services, databases and statistics	300
	POSS evaluators	To define by a SLA Operation of a scientific committee Opportunity and Quality	150
	POSS Controllers	To define by a SLA Users Forums Mailing lists Content of DBs Newsletters	150
	Deliverance of legal services	To define by a SLA Maintaining the contractual framework; Advice on law evolutions; Assist members in legal issues (patents etc.)	150
	Translations	To define by a SLA Translate software descriptions, main pages and contract modifications, and other services (to fix)	150
	Quality Control	Check the respect of SLA Maintain scoreboards Report monthly	75

Funding the POSS

As public service, the POSS will be available to Public sector members. Members may contribute to this new service by paying a contribution, but at the early beginning the number and quality of the software specifically or exclusively distributed by the new service will be reduced.
Paying for a non existent – future service is not realistic: just as when marketing a data bank (for example a legal data bank like Celex or Eur-Lex) the critical mass must be achieved in advance in order to get contributions. The main effort expected from public sector members will be to contribute their software, according to the POSS quality requirements.

Trying to obtain alternative revenues from "banners" or advertisement is also a non-sense for a "non great public oriented" service as the POSS and would damage its image of serious.

Private sponsorship (some large IT corporate invest massive amounts in open source technology) is not the POSS way: It would damage its neutrality. All private sector investments are strongly commercially oriented (for example an IBM-HP consortium will support only Linux application, as the public sector community may like to promote also MS/Windows running OSS applications)

Expecting money from service providers is not realistic in a first stage: service provider interest will come once the POSS success provides them a return for this investment, not at first stage. The potential service providers interest may reside in being referred in providers lists (as it is done in the Berlios and Atica sites)

In conclusion, at least during the first five years period, an exclusive funding by European Institutions seems the only possible solution.

The public funding of open source initiatives was debated at several occasions[91] in US as well as in EU.

Based on the idea that open source software has such public benefits and may be used as public platform for e-government services, a frequently taken position is that a government should promote its use through funding or regulation[92]. At the same time, we noted that representatives of the software industry claims that the market only should decide and it would not be fair to favour OSS. Open source advocates as Eric Raymond have the same opinion for other reasons: they are basically hostile to all public interventionism and believe the OSS Bazaar[93] will succeed without any help, due to its own quality.

[91] See references in part 3 of the Study into the use of Open source software -
[92] See on this theme, the roundtable discussion issued by The American Prospect – TAP – between Eric Raymond, Nathan Newman, Jeff A Taylor and Jonathan Band (http://www.prospect.org/controversy/open_source/)
[93] Reference to the famous E. Raymond book "The Cathedral and the Bazaar" describing how the anarchic cooperative OSS development model (the Bazaar) had proven its capability to match the organized, planned and hierarchic software development strategies of IT giants (the Cathedrals).

But the aim of the POSS is not to promote open source against proprietary software: it is just to optimise public investments by sharing public administrations developed software. For doing that, the software must be licensed, adapted and redistributed, and it appears that the convenient licenses to respond to the needs are the "Open source" ones.

Existing Public sector funding examples are:

- Knowledge sharing services. For example, the Canadian government started (in 1988 already) a Software Exchange Service (SES), in order to redistribute software surplus (including the proprietary licenses "stock") and existing public sector applications.
  The SES has known a constant expansion, working with more than 10.000 clients and over 200 organisations.
  The global saved amount is difficult to estimate, but for a single application, the "Salary Management System" the savings where estimated at 17.000.000 Can$[94] (this is to compare with the yearly SES service cost, estimated at 800.000 Can$, entirely funded by the government budget).
  At the end of 2002, the SES service will be changed into a web portal: the KES or "Knowledge Exchange Service" (note the move from "Software" to "Knowledge"). This move will raise up the "Open source" issue and the opening to a wider range of beneficiaries (e.g. emerging countries).
- Direct support of projects by funding open source developers (this way is followed e.g. within the European IST 5th framework programme and by the French government, reserving a part of the FNRS funding for open source projects).
  As the POSS project is innovative in itself (creating an original pan-European development and legal framework) a part of the funding may be found there.
- EU Programmes facilitating the European integration (e.g. Phare for EU candidates in order to facilitate the take over of the "acquis"): acquiring best practices in eGovernment and other public sector IT applications may enter in the scope of such programmes.
- EU programmes supporting emerging countries (Europaid).

[94] As reported into the "Give and Take" 15th anniversary newsletter – 26th April 2002 – of the Knowledge Exchange Service of the PWGSC (Public Works and Government Services Canada) – info.knowledgeXange@pwgsc.gc.ca

Conclusion about financing the POSS

When speaking about open source (or Free Software) a common temptation is to ask if all or part of the POSS service cannot be elaborated “for free” or set up as an open source project delivered by volunteers.
This may be so, but it could result in an unofficial project, with no administration, no SLA, no legal security, no neutrality. In best case it will them present the advantages of an enthusiastic community (as there are many sites and portals of free users associations in France, Spain, Germany etc.), but it will not appear as an official European Commission contribution, with advantages and constraints of such an organisation.

An alternate solution, sponsoring an existing national organisation or a private firm initiative also presents constraints and lack of neutrality. Our mission was not to enter in that process.

As we have declared many times, free software does not mean “for free”. The POSS is a complex organisation that will require a serious investment and commitment from the European Union itself, at least during the first five years.

If, after five years, the POSS is a success and has demonstrated (with statistical and quality control evidence) a public utility, then it will attract more volunteers and contributors, or even sponsors and the financing policy may be reviewed

We have to consider that in administrations the idea to re-use software developed by others is extremely new. 2001 was the first year when public sector administration really started to “benchmark” their respective solutions[95]. Constructing a Pooling Portal based on the assumption it will be progressively used by a growing number of public administrations is still a bet today: in large eGovernment contracts that are attributed now (in Government portals, PKI, private networks, development or monitoring of Phare projects etc.) the actual tendency is still to act on a strictly national (or even regional) basis, and to duplicate efforts, even inside the same country.
Therefore, benefits will not be immediate: return on investment and cost savings related to the re-use of shared software will take at least that time (5 years) to be demonstrated.

[95] As in the eGovernment conference, organized by the Belgian Presidency in November 2001

Conclusion

The notions.

The title of the feasibility study “Pooling Open source Software” (POSS) is a mix of three notions:
Pooling is the main idea: not reinventing the wheel, sharing with other, making significant sparing and obtaining the best value for money;
Software is important too, but it does not cover all the scope of the requirements: a service that covers more than just software (code or binary), but the sharing of **knowledge and competences**: ideas, news and events, links to other organisations, advices etc.
Last “**Open source**”: the concept is more confusing, as it refers to a variety of licenses. It is clear that generally speaking, the pan-European re-use of public sector software will require code adaptations to specific needs and re-distribution to new users. These two characteristics are common to open source licenses. It is clear also that development speed, quality and security will be optimised with the creation of developer’s communities where little groups or individuals can contribute informally and release modules through appropriate Internet tools. For all these reasons, as demonstrated in Chapter 1, a reference to the open source model is appropriate.

However, in the practice, much public sector software is running on proprietary platforms or cannot run without proprietary components (e.g. a fleet management solution based in an Oracle or MS Access database). On the other hands, the contributors may require to give some of their products according to specific licenses that are not in full conformity to the Open source Initiative requirements.
As these solutions should not be excluded from the service, the POSS name may be an issue and the service may also be named with (for example): “PKS - Pooling Knowledge Service” “IDA European Competence Center” or KES “Knowledge Exchange Service”[96]

The legal framework

The POSS contractual framework includes:

- The general terms of the pooling service (what we call the POSS chart);

[96] The Canadian government has implemented in 1988 already a SES (Software Exchange Service) that will be – for the same reasons – changed into a KES (Knowledge Exchange Service) in the future Web Portal (end of 2002)

- The contract between the author and the POSS (what we call the mandate or “commission”, as the POSS will represent the author – licensor when contracting with the user - licensee);
- Specific agreements related to liability, competent judge and applicable law, patent issues;
- The license agreement itself that should be selected by the author – licensor (and accepted by the user – licensee).
- Service level agreements with various POSS service actors, and among them, one or more legal advisers to help candidate licensors to solve licensing (copyright) and patent problems, as each case is or may be specific (no general – all purpose – answer exists)

Concerning the beneficiaries of the POSS, it should be limited to registered users (public sector in a flexible sense). In a first stage (observation period) user registration may be limited to European public sector. In a second stage, the POSS can bring a substantial contribution to emerging countries IT development (e.g. in Africa, Asia) depending on European authorities decision.

Limitations should not concern the end user itself and a “specific European public sector license” should not be promoted: it is difficult for legal reasons (we will see that, due to the “copyleft” effect, original OSS licenses as the GPL and MPL must be used without modifications in some conditions) and also for practical reasons: an “exotic” license will not help to create a developers community and the end user control is difficult in an open source environment allowing re-distribution.

The contractual trail process with all actors must be carefully registered (licensee authentication, time stamp, contract archiving) in order to produce contractual evidences if needed.

Some POSS service (news, forum, etc.) should be accessed by anybody, but member authentication should be required for specific services as downloading.

The possible sub-sequent software redistribution by initial licensees is depending on the license and will not be systematically registered or controlled. It will stay outside the POSS scope and liability if any, as other operations done from sites that will be just “referred” by the POSS by URL links.

Concerning the choice of the licenses, this is the **sole responsibility of the licensor**. The POSS may help or propose simplified choices between 3 licenses:

- BSD to give the software to all users and to “**all developers community**” without any restrictions or discrimination regarding the use (from the same original contribution some licensee may develop proprietary software, and others may continue with Open source);
- GPL to give the software to all users and to the “**free software developers community**”;

- MPL variant if for any reasons some components must stay proprietary or if the initial licensor wants to benefit from a close follow-up of all code modifications, and reduce “piracy” (meaning there the simple duplication of binary package, without contribution or added value).

The licensor must be free to select or to elaborate another license, as long it is compatible with the POSS chart (general terms).

According to the subject (public sector specific software) and the restricted downloading access, the risk of competition with the private sector is reduced: the POSS will liberate budgets for more quality and integration. These services may be delivered by private partners as usual.
Where the BSD license is selected, the POSS will also provide proprietary software development opportunities to the software industry (without impeaching other developers to start from the same code to deliver other types of open versions).
One of the conditions of the “non-competition” is the POSS neutrality regarding platforms and possible use of proprietary components (e.g. some public sector solutions require the use of an Oracle or MS/Access database).

The requirements

To address the needs (Chapter 3), the POSS should include a series of services:

- A multilingual portal with a clear site map, pan-European content and standard data format;
- Registered members management;
- Software submission, description, classification and downloading management…);
- Links to national initiatives and addition of any other link related to projects (software) and members;
- Library management to attach various documents related to software (technical documentations, user guides, knowledge and competences regarding an European framework of open standards);
- News, Forums and mailing lists;
- Opinion surveys
- Software hosting for downloading (FTP) directly from the POSS
- A scientific committee. Even if no formal guarantee or quality audit will be possible, the conformity to formal quality criteria (documentation, manuals) and available test scenarios, as the advices of experts corresponds to a frequently expressed need.
- Legal advisory service, and contractual framework management (including the registration or log of all contracted downloads, by authenticated users, software, time stamps, agreed license types).

The Technical design

All tools required to construct and deploy the POSS have been identified in chapter 4.

- The definition of the technical framework of standards to respect when designing the POSS. This part presents the general site design and the security features.
- A list of possible tools requested to set up the POSS. This list is based on tools used by other similar sites and that are proved to be compatible (Operating system, Web and FTP servers, Database, e-Mail, Statistics, Link checker, Scripting language).

  The tools presented as ***examples*** in the feasibility study are "Open source" (because mature OSS tools are available and to avoid confusion and debate that should damage the credibility of the Study). However, sharing public sector specific software thanks to open source licenses could as well be done – technically – by using one or more proprietary components if justified by consistency, global integration and delivery costs or performances.

The Maintenance

The maintenance represent the most important part of a long-term investment, and is based on the following service roles:

- System administrator
- Evaluator (of software)
- Controller (of daily POSS content, users, news, forums)
- Software (or project) responsible
- Developer
- User
- (external) Legal adviser

The maintenance process include:

- **Components**
  POSS hardware and software as the web server, the forum tool, the e-mail infrastructure, the link checker tool;
- **Content**
  maintenance of software index and home pages, links, news, and library indices;

- **Services**
  member’s registration, legal advice and framework, forums, mailing list, surveys, statistics.

The Costs

The POSS is not (= should not be) a software, even if it is made of software. It should be a service.
A quality service will require the same level of investment as any other pan-European service. If this service is supported or even organized by the European Commission, the constraints concerning the quality, the number of languages, the availability and the neutrality will be high and must be delivered based on various service level agreements (SLAs) with professional organisations.

The budget includes an initial investment (prototype) between 510.000 and 1.060.000 euros, a deployment of 340.000 euros and 5 years of operations at 975.000 euro per year, bringing the TCO at about 6 millions on five years.

The return on investment will not be immediate, as the initial effort to evaluate and “adapt” existing software to open source pooling will be important and as new projects take time to gain their maturity.

During this initial five years period, a public POSS funding is required. No funding by initial beneficiaries is realistic: their first role is to provide software. The large “Hardware and Services” IT industry is not interested, as their ambition is to concentrate efforts on specific applications that are not compatible with the POSS neutrality.

*
*   *

Bibliography

- Jaeger/Metzger, Open source Software – Rechtliche Rahmenbedingungen der Freien Software – Verlag C.H. Beck Munchen 2002
- La Licence Publique Générale GNU, Mémoire de DEA par Mélanie Clément-Fontaine, Université Montpellier I (laboratoire CNRS ERCIM), 1999.
- Le "Copyleft" ou l'état des interrogations quant à l'impact des NTIC en tant qu'élément déstabilisateur des règles de propriété intellectuelle, David Geraud, 12 novembre 1999.
- Case study of a non open source license: SCSL: a study of the Sun Community Source License (SCSL). In « Free Software / Open source: Information Society Opportunities for Europe? », Working group on Libre Software (Information Society Directorate General of the European Commission), Version 1.2, April 2000.
- Kelm, Stefan. The PKI page. © *2000-2002*. 15 May 2002. 31 May 2002 <*http://www.pki-page.info/* >
- Bray, Brian. The Spirit Project. © 2001 Minoru Development SARL, Conecta srl, and/or Sistema Information Systems. 31 May 2002. 31 May 2002 <http://www.euspirit.org>
- Price, Derek. Concurrent Versions System, the open standard for version control. Copyright © 1999-2002 CollabNet, Inc. 17 April 2002. 3 June 2002 <http://www.cvshome.org>
- Bray, Brian. Minoru development corporation. © 1997-2000 Minoru Development Corporation. 1 February 2002. 3 June 2002 <http://www.minoru-development.com>
- Greenman , David. FreeBSD, the power to serve. © 1995-2002 The FreeBSD Project. 3 June 2002. 3 June 2002 <http://www.freebsd.org>
- The Shmoo Group. 24 April 2002. 3 June 2002 <http://www.shmoo.com/securecode/>
- Sourceforge.net, Breaking Down the Barriers to Open source Development. © Copyright 2002 - OSDN Open source Development Network, All Rights Reserved. 3 June 2002. 3 June 2002 <http://sourceforge.net >
- Roessler, Thomas and Köhntopp, Kristian. OSS funding. 3 June 2002 <http://www.koehntopp.de/kris/artikel/oss_funding/>
- Stallman , Richard. GNU's Not Unix. Copyright (C) 1996, 1997, 1998, 1999, 2000, 2001 Free Software Foundation, Inc. 24 May 2002. 3 June 2002 <http://www.gnu.org, http://www.fsf.org>

- Bdale, Garbee. Debian GNU/Linux, the Universal Operating System. Copyright © 1997-2002 SPI. 14 January 2002. 3 June 2002 <http://www.debian.org>
- Mallett , Steve. The Open source Initiative. Copyright © 2001 by the Open source Initiative. 28 November 2001. 3 June 2002 <http://www.opensource.org>
- X.org. 3 June 2002 <http://www.x.org>
- The Apache Software Foundation. Copyright © 1999-2002, The Apache Software Foundation. UPDATE. 3 June 2002 <http://www.apache.org>
- Welcome to Cryptix. Copyright © 1995-2000 The Cryptix Foundation Limited. All rights reserved. 26 August 2001. 3 June 2002 <http://www.cryptix.org>
- Fallon, Henry. World Wide Web Consortium. Copyright © 1994-2002 W3C® (MIT, INRIA, Keio), All Rights Reserved. 31 May 2002. 3 June 2002 <http://www.w3.org>
- Python language website. 3 June 2002 <http://www.python.org>
- Welcome to Zope.org © 2002 Zope Corporation All rights reserved. 3 June 2002 <http://www.zope.org>
- Zeilenga , Kurt. OpenLDAP. Copyright 2002, OpenLDAP Foundation. UPDATE. 3 June 2002 <http://www.openldap.org>
- Phorum.org. 3 June 2002 <http://www.phorum.org>
- MSDN Home. © 2002 Microsoft Corporation. All rights reserved 3 June 2002. 3 June 2002 <http://msdn.microsoft.com>
- Samba web pages. 3 June 2002 <http://us1.samba.org>
- Wilcox-O'Hearn, Bryce. Zooko introduction. COPYRIGHT. UPDATE. 3 June 2002 <http://www.zooko.com>
- Open Content. 3 June 2002 <http://www.opencontent.org>
- INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE. © INRIA. 13 May 2002. 3 June 2002 <http://www.inria.fr>
- The Linux Home page at Linux online. © Linux Online Inc. 27 May 2002. 3 June 2002 <http://www.linux.org>
- Welcome to Freshmeat.net. © Copyright 2002 OSDN Open source Development Network, All Rights Reserved.3 June 2002. 3 June 2002 <http://freshmeat.net/>
- Agence pour les Technologies de l'Information et de la Communication dans l'Administration. 30 May 2002. 3 June 2002 <http://www.atica.pm.gouv.fr/>
- BerliOS, Der Open-Source-Mediator. Copyright ©2000 FOKUS. UPDATE. 3 June 2002 <http://www.berlios.de/>
- The source of Java ™ Technology. Copyright © 1995-2002 Sun Microsystems, Inc. 3 June 2002 <http://java.sun.com>
- Engelschall, Ralf. The Apache Interface to OpenSSL. Copyright © 1998-2001 Ralf S. Engelschall, All rights reserved. 27 March 2002. 3 June 2002 <http://www.modssl.org/>
- Engelschall, Ralf. The Open source toolkit for SSL/TLS. Copyright © 1999 The OpenSSL Project, All rights reserved. UPDATE. 3 June 2002 <http://www.openssl.org/>

- Hudson, T J. SSLeay Documentation. 3 June 2002 <http://www.columbia.edu/~ariel/ssleay/>
- MySQL. © 1995-2002 MySQL AB. 28 May 2002. 3 June 2002 <http://www.mysql.com/>
- Building dynamic websites with PHP. Copyright © 2001 Maguma AG. 3 June 2002 <http://phpwizard.net>
- Sendmail Home Page. 3 June 2002 <http://www.sendmail.org/>
- Metheringham, Nigel. Exim Internet Mailer. 26 March 2002. 3 June 2002 <http://www.exim.org/>
- Mailman, the GNU mailing list manager . © 1998,1999,2000,2001,2002 Free Software Foundation, Inc. 3 June 2002 <http://www.list.org/>
- Raymond , Eric S. FetchMail home page 3 June 2002 <http://www.tuxedo.org/~esr/fetchmail>
- Bradford L., Barrett. Home of the Webalizer. 25 December 2001. 3 June 2002 <http://www.webalizer.com>
- Mozilla.org. Copyright © 1998-2002 The Mozilla Organization. 3 June 2002. 3 June 2002 <http://www.mozilla.org/>
- linuxhelp.dyndns.org. 3 June 2002 <http://linuxhelp.dyndns.org/>
- Bowlin , James B.. Linklint - fast html link checker. © Copyright 1997 - 2001 James B. Bowlin. 8 August 2001. 3 June 2002 <http://www.mindspring.com/~bowlin/linklint/>
- PHP: Hypertext Preprocessor. Copyright © 2001, 2002 The PHP Group. 3 June 2002. 3 June 2002 <http://www.php.net/>
- OPENSSH. Copyright © 1999-2002 OpenBSD. 23 May 2002. 3 June 2002 <http://www.openssh.com/>
- Wheller, David. SCLOCCount. 3 June 2002 <http://www.dwheeler.com/sloccount/>
- User Authentication With Apache And PHP. copyright Melonfire 2000-2002. All rights reserved. 13 March 2002. 3 June 2002 <http://www.devshed.com/Server_Side/PHP/UserAuth/page1.html>
- Fermigier, Stéphane. Linux Center: index general. Linux Center © 1997-2001. UPDATE. 3 June 2002 <http://www.linux-center.org>