

 <p>FLOSSMetrics project Free/Libre and Open Source Software Metrics</p>   <p>Sponsored through Framework Programme Sixth (Call 5) by the European Commission</p>		Document Information Version: 1.0 Date : Oct. 10 2007 revision: 0
		Owning Partner: Conecta
		Author(s): Carlo Daffara
		Reviewer(s): Jesus M. Gonzalez-Barahona
		To: PUBLIC
		Purpose of distribution: Final Version
The FLOSSMetrics Consortium consists of: Universidad Rey Juan Carlos, University of Maastrich, Wirtschaftsuniversitaet Wien, Aristotle University of Thessaloniki, Conecta s.r.l., Zea Partners and Philips Medical Systems PMS Nederland B.V.		Printed on at
Status: <input type="checkbox"/> Draft <input type="checkbox"/> To be reviewed <input type="checkbox"/> Proposal <input checked="" type="checkbox"/> Final/Released	Confidentiality: <input checked="" type="checkbox"/> Public - Intended for public use <input type="checkbox"/> Restricted - Intended for FLOSSMETRICS consortium only <input type="checkbox"/> Confidential - Intended for individual partner only	
Deliverable ID:	D8.1.1	
Title:	<p style="text-align: center;">Guide for SMEs</p>	
License for distribution: This work is licensed under a Creative Commons Attribution-Share Alike 2.5 License . (The license can be found in http://creativecommons.org/licenses/by-sa/2.5/) The original version of this document is available at http://flossmetrics.org		




Deliverable: D8.1.1

Title: Guide for SMEs

Executive Summary:

FLOSS (free, libre, open source software) is one of the most important trends in IT since the advent of the PC and commodity software. However, despite the potential impact on European firms, its adoption is still hampered by limited knowledge, especially among SMEs that could potentially benefit the most from it. This document presents a set of guidelines and suggestions for the adoption of FLOSS within SMEs, using a ladder model that will guide companies from the initial selection and adoption of products for their IT infrastructure up to the creation of suitable business models based on FLOSS.

	Guide for SMEs Deliverable ID: D8.1.1	Page : 3 of 82
		Version: 1.0
		Date: Oct. 10 2007
		Status : Final Confid : Public

CHANGE LOG


Ver.	Date	Author	Description
0.1	19/07/2007	Carlo Daffara	Initial proposal
0.2	14/08/2006	Carlo Daffara	Improvements in the "What is FLOSS" chapter with data on collaborative models, a simplified licensing view, mention on non-code contributions in projects. New chapter "Myths on FLOSS". "Businesses Models" section modified. New bibliography and appendix added.
0.3	23/08/2007	Carlo Daffara	New chapters and appendix added.
0.4	25/08/2007	Carlo Daffara	Added executive summary and some modifications on the style.
0.5	19/09/2007	Jesus M. Gonzalez-Barahona Santiago Dueñas	FLOSSMetrics deliverable format. Full review, minor fixes.
1.0	10/10/2007	Jesus M. Gonzalez-Barahona	Full review, publishable version

APPLICABLE DOCUMENT LIST


Ref.	Title, author, source, date, status	Deliverable Identification

TABLE OF CONTENTS

Introduction	6
1. What's Free/Libre/Open Source Software?.....	7
FLOSS as a licensing model.....	8
FLOSS as a development model.....	11
2. Ten myths about free/libre open source software.....	15
Myth #1: It's a Linux-vs-Windows thing.....	15
Myth #2: FLOSS is not reliable or supported.....	16
Myth #3: Big companies don't use FLOSS.....	19
Myth #4: FLOSS is hostile to intellectual property.....	19
Myth #5: FLOSS is all about licenses.....	21
Myth #6: If I give away my software to the FLOSS community, thousands of developers will suddenly start working for me for nothing.....	21
Myth #7: FLOSS only matters to programmers, since most users never look under the hood anyway.....	21
Myth #8: There is no money to be made on FLOSS.....	22
Myth #9: The FLOSS movement isn't sustainable, since people will stop developing free software once they see others making lots of money from their efforts.....	24
Myth #10: FLOSS is playing catch-up to Microsoft and the commercial world.....	24
3. Basic FLOSS adoption models.....	26
The FLOSS adoption ladder.....	27
4. Finding and selecting software.....	30
5. Best practices for FLOSS adoption.....	38
Management guidelines.....	38
Be sure of management commitment to the transition	39
Prepare a clear overview of what is expected from the migration or adoption, including measurable benchmarks.....	39
Make sure that the timetable is realistic.....	40
Review the current software/IT procurement and development procedure	40
Seek out advice or search for information on similar transitions.....	40
Avoid "big switch" transition, and favor incremental migrations.....	41
Assign at least a person to interacting with the OSS community or the OSS vendor, and try to find online information sources.....	42
Technical guidelines.....	42
Understand the way OSS is developed.....	42
Create a complete survey of software and hardware that will be affected by the migration, and what functionality the company is looking for	43
Use the flexibility of OSS to create local adaptations.....	44
There is much more software available than what is installed by default.....	44
In selecting packages, always favor stability over functionality.....	45
Design the workflow support infrastructure to reduce the number of "impedance mismatches"	45
Introduce a trouble ticket system.....	45
Compile and update a detailed migration workbook	45
Social guidelines.....	46
Provide background information on OSS.....	46

	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 5 of 82
		Version: 1.0
		Date: Oct. 10 2007
		Status : Final Confid : Public


Don't force the change on the users, provide explanations.....	46
Use the migration as an occasion to improve users skill.....	47
Make it easy to experiment and learn.....	47
6. <u>FLOSS-based business models</u>.....	49
Externally funded ventures	50
"Needed improvement" funding	50
Indirect funding / Loss-leader.....	52
Internal use	52
"Best knowledge here" without constraints	53
"Best knowledge here" with constraints	53
"Best code here" without constraints	54
"Best code here" with constraints/Time-decaying licenses.....	54
Dual licensing.....	54
Unfunded developments	55
Specialized Service-based business models.....	55
Software selection support	56
Installation support.....	57
Integration support	57
Technical suitability certification.....	58
Legal certification.....	58
Training.....	59
Ongoing maintenance and support contracts.....	60
Migration services.....	61
Mediation services.....	62
Custom development.....	63
Assessment of FLOSS business models usage.....	63
<u>Bibliography</u>.....	69
<u>Appendix 1: estimating the number of active FLOSS projects</u>.....	74
<u>Appendix 2: QSOS assessment score tables</u>.....	77

	Guide for SMEs Deliverable ID: D8.1.1	Page : 6 of 82
		Version: 1.0 Date: Oct. 10 2007
		Status : Final Confid : Public

Introduction


"Open source software is the most significant all-encompassing and long-term trend that the software industry has seen since the early 1980s". This is one of the conclusions of a recent IDC report [IDC 06], which shows how much the perception of FLOSS (free, libre, open source software) has changed in the recent years. Right now, the majority of developers in the world are using FLOSS [Forr 07], and FLOSS platforms are used in one way or another by a large share of companies.

Despite this situation, there is still a significant barrier in the adoption process for small and medium companies, both in terms of using FLOSS internally and in creating products and services based on FLOSS products. The purpose of this report is to provide a simple and in-depth view of the fundamental aspects of FLOSS, how to adopt it within a small/medium company, and how to build a sustainable business based on it.

	Guide for SMEs Deliverable ID: D8.1.1	Page : 7 of 82
		Version: 1.0 Date: Oct. 10 2007
		Status : Final Confid : Public

1. What's Free/Libre/Open Source Software?

It may be a surprise to discover that the software market that we take for granted, based on the idea of "shrink-wrapped" packages that are easy to buy directly by the user, is relatively recent. In the beginning, software was bundled with hardware by the manufacturer. Due to the complexity and cost of development (and the relatively limited power of those first computers), to the business models of the manufacturers (based on selling hardware), and to other factors, users freely shared source code and advice, in a collaborative way that led to the creation of user groups like SHARE (Society to Help Avoid Redundant Efforts, founded in 1955 and centered around IBM systems), and DECUS (for Digital Equipment computers, and later for HP systems), both still alive. Code was also commonly shared in academic journals, like the famous "Algorithms" column of the "Communications of the ACM" journal.

	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 8 of 82
		Version: 1.0
		Date: Oct. 10 2007
		Status : Final Confid : Public


With the "unbundling" process (the separation of hardware and software catalogues), the first "packaged" software products appeared on the market in the 1970s. With the advent of the first personal computers (the Apple II, the IBM PC and many others) the shrink-wrapped software market became the most familiar to users, being still today a significant part of the overall IT landscape. It is important however to notice that such market represents only around 25% of the total value of the software market, with the remaining composed of custom software developed under contract and software developed in-house [OECD 02].

FLOSS¹ as a licensing model

Building on a tradition laid by academic institutions like MIT, Richard Stallman founded in 1983 the Free Software Foundation (FSF) to find a way to preserve the freedom of users to study, understand and modify software, in direct link with the hacker culture of openness and sharing of information. The objective of the FSF was to create a complete reimplementation of the Unix operating system, at that time an important reference for most large companies and research centres. With this purpose Stallman and many others created a complete development and execution environment, for which in the late 1980s the kernel (the underlying core of an operating system) was the only missing component. This gap was filled soon, in 1991, by two different teams: the effort led by Linus Torvalds developed the Linux kernel, while William and Lenny Lott wrote a series in the Dr. Hobbs Journal on how to port BSD Unix to i386-based PCs, creating the basis for a complete, free operating system for modern personal computers [DB 00].

The Free Software Foundation places a strict emphasis on the underlying "four freedoms":

¹Richard Stallman and the FSF introduced the term "free software". Later, the Open Source Initiative proposed "open source software", allegedly to avoid the linguistic uncertainty associated with the English term "free", specifically used by the Free Software Foundation to preserve the underlying concept of freedom. The "libre software" term was introduced for the same reason, and used specially in Europe. The term "FLOSS" was introduced by Rishab Gosh in the context of EU-funded project "Free/Libre and Open source software: survey and study" started in 2002 as a catch-all term for free software and open source as described in this section. In this report we will use mainly the term FLOSS.

	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 9 of 82
		Version: 1.0
		Date: Oct. 10 2007
		Status : Final Confid : Public


- The freedom to run the program, for any purpose (freedom 0)
- The freedom to study how the program works, and adapt it to your needs (freedom 1). Access to the source code is a precondition for this
- The freedom to redistribute copies so you can help your neighbour (freedom 2)
- The freedom to improve the program, and release your improvements to the public, so that the whole community benefits (freedom 3). Access to the source code is a precondition for this.

For this reason, the FSF created a set of "free software licenses", and among them the GPL (general public license) and LGPL (lesser general public license) that are the most widely used, both in terms of number of projects and in number of lines of code covered.

Unfortunately, in many situations the term "free software" is frequently interpreted as "gratis", that is, with no price; a fact that forced the FSF to introduce the slogan "free as in free speech, not as in free beer". The free software environment moved at a significant pace, up to the development of complete user environments such as GNOME and KDE, and to the design in 1998 of the "open source" trademark, created to present a more pragmatic alternative to the somewhat "political" orientations of the FSF. The Open Source definition is based on a similar set of conditions:

***“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 programs from several different sources. The license shall not require a royalty or other fee for such sale.*

***Source Code** The program 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 program. Deliberately obfuscated*

	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 10 of 82
		Version: 1.0
		Date: Oct. 10 2007
		Status : Final Confid : Public

source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed.

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.*

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 program 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.*


No Discrimination Against Persons or Groups *The license must not discriminate against any person or group of persons.*

No Discrimination Against Fields of Endeavour *The license must not restrict anyone from making use of the program in a specific field of endeavour. For example, it may not restrict the program from being used in a business, or from being used for genetic research.*

Distribution of License *The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.*

License Must Not Be Specific to a Product *The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.*

License Must Not Restrict 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 programs distributed on the same medium must be open-source*

	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 11 of 82
		Version: 1.0 Date: Oct. 10 2007
		Status : Final Confid : Public

software.

License Must Be Technology-Neutral *No provision of the license may be predicated on any individual technology or style of interface.”*


Both groups maintain a list of licenses that comply with the terms of the Free Software Definition, or the list of conditions for using the term "open source". In fact, there are more than 50 licenses identified as "open source" or "free software", but fortunately they can be classified in a very simple way as [Sun 06, UU 05]:

- *"provide credit"*: use, modification, redistribution are allowed, but credit to the original author is due, if redistributed. Examples: BSD license, Apache License v2.
- *"provide fixes"*: use, modification, redistribution are allowed, but source code for any changes must be provided to the original author, if redistributed. Examples: Mozilla-style licenses (Mozilla Public License).
- *"provide all"*: use, modification, redistribution are allowed, but source code of any derived product must be provided, if redistributed. Example: GPL.

When code from different projects is mixed and redistributed, the issue of license compatibility becomes important. An extremely detailed matrix with licensing compatibility with regards of GPL (including the recently released GPLv3 license) is available at [Fed 07]; in any case, whenever a product is released or distributed, it is advisable to ask advice of an attorney with expertise in FLOSS licenses and intellectual property (a similar advice applies to proprietary software releases).

FLOSS as a development model

While FLOSS as a definition covers exclusively the licensing regime, by extension the “openness” of the code introduced the possibility of sharing development efforts among different groups, in a way similar to those of the early user groups of the sixties. In this sense, Eric Raymond introduced in his seminal paper “The cathedral and the

	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 12 of 82
		Version: 1.0
		Date: Oct. 10 2007
		Status : Final Confid : Public

bazaar” the concept of shared development, contrasting this “bazaar” style where every developer is free to choose on what part of the code to work, in contrast to the “cathedral” or formalized development approach that is rigid and structured [Ram 00].

While the concept took hold quickly, the reality is that collaboratively developed projects tend to be executed in a continuum between cathedral and bazaar; for example, for most projects there is a formal structure (with many sub-projects, more open to external contributions) while other are strictly formal (for example, projects that use FLOSS code in a certified environment, such as avionics or safety-critical systems). The important point raised by Raymond is the fact that both coding and ancillary activities like bug fixing and production of documentation can be shared in a large community, creating in a sense "virtual software houses" that in a voluntarism way provide effort and resources; this helps also in the leverage of a large community of expert users, that can contribute back in a significant way, as shown in [VH 03, VH 05].

When such collaboration takes place, it may be not only in the form of source code, as for example is commented in [July 06]: *“In the year 2000, fifty outside contributors to Open Cascade provided various kinds of assistance: transferring software to other systems (IRIX 64 bits, Alpha OSF), correcting defects (memory leaks...) and translating the tutorial into Spanish, etc. Currently, there are seventy active contributors and the objective is to reach one hundred. These outside contributions are significant. Open Cascade estimates that they represent about 20 % of the value of the software.”*

A similar view has been presented in [Seri 06], where one of the leaders of the KDE project² presented the elements that collectively contribute to KDE:

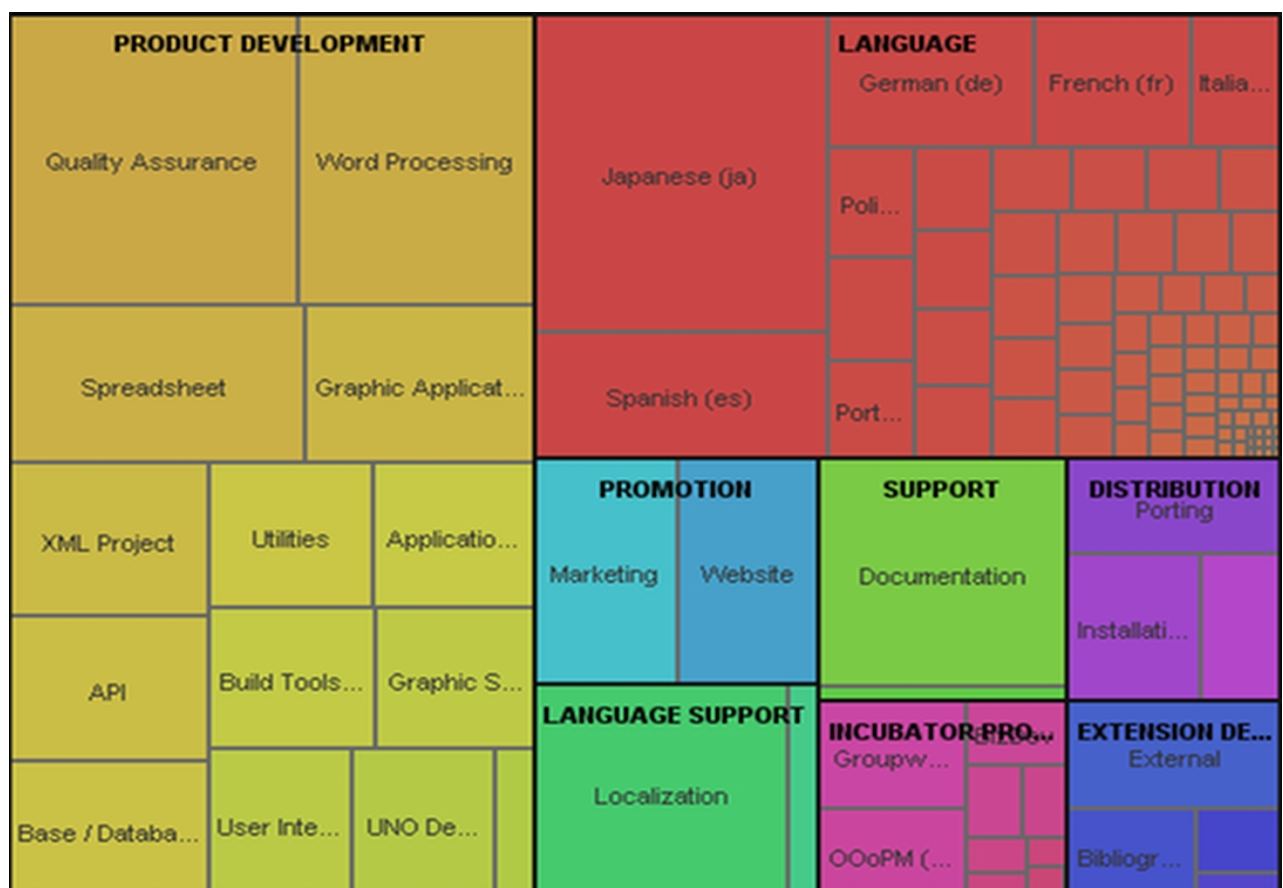
- Artwork
- Documentation
- Human-computer interaction

²KDE is a complete user desktop environment, created originally as a libre alternative of the Unix CDE environment, and later evolved to encompass libraries, end-user software and training material.


- Marketing
- Quality Assurance
- Software Development
- Translation

If overall software suitability to the task is considered, it is clear that non-code contributions are as important as source code. For example translations, documentation and overall quality are vital for the software to be adopted by end-users worldwide.

Another example comes from [Sue 07], where the number of participants within individual OpenOffice sub projects were counted:




As it can be inferred from the area graph, there are roughly as much non-code contributors than those working on product development and related projects (that are directly related to source code).

	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 14 of 82
		Version: 1.0 Date: Oct. 10 2007
		Status : Final Confid : Public

This form of collaboration can happen even between competing companies. For example, news about potential security vulnerabilities are commonly shared among different competing Linux vendors. As an example, Mark Cox of Red Hat (a widely used distribution of Linux) analysed the results of two years of incident responses, and found that the largest share of information was coming from the other peer FLOSS distributors [Cox 07].

In more recent years, companies started the adoption of this collaborative model to develop software and services, sometimes supplementing the volunteer communities and sometimes starting new projects and providing substantial resources to its continuation. This later stage (the commercialization stage) is more focused on the sustainability of business models adopted by said companies, and is the main focus of chapter 6.


	Guide for SMEs Deliverable ID: D8.1.1	Page : 15 of 82
		Version: 1.0 Date: Oct. 10 2007
		Status : Final Confid : Public

2. Ten myths about free/libre open source software

In 1999, Tim O'Reilly, founder of a popular open source-oriented publishing house, gave a keynote speech to an audience of Fortune 500 executives called "ten myths about open source software". As those myths are still perceived today, as shown by recent reports [CIO 07, ED 05, Forr 07], and are still perceived as a barrier towards FLOSS adoption, we will try to provide here a SME-oriented and pragmatic answer to all of them.

Myth #1: It's a Linux-vs-Windows thing.

Most recent debates about FLOSS were focused on an all-or-nothing perception. For example, when introducing FLOSS in a company, a full software migration is often considered as necessary. This, and the fact


	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 16 of 82
		Version: 1.0
		Date: Oct. 10 2007
		Status : Final Confid : Public

that there is limited knowledge of FLOSS projects except for a few very widely known ones (like Linux, Apache, OpenOffice.org), created the perception that most FLOSS is designed and targeted as a direct competitor of Microsoft products. The reality is that there is an enormous number of active projects in practically any IT field, including business-specific (such as ERP systems), being most of them cross-platform, capable of running Microsoft Windows, Apple's OSX (which is itself based on more than 300 open source projects) or Linux. As can be found in Appendix 1, there are more than 18,000 FLOSS projects that are stable and mature for adoption by SMEs.

Myth #2: FLOSS is not reliable or supported.

This myth is based on a common perception that FLOSS is exclusively developed by volunteers in a non-coordinated or unstructured way. There are many errors in this view:

- **the volunteer perception:** while volunteer contributions are a significant part (and sometimes the majority) of large scale projects, around 50% of developers have received a financial compensation for working on FLOSS projects, either directly paid to improve the projects or paid to support them. This has been shown in recent studies [Gosh 05, Gosh 06] and can be inferred directly by the fact that in the software industry at large, 68% of software products include directly FLOSS-derived code.
- **paid programmers are better:** even for the percentage of contributions that are coming from volunteers, it is commonly perceived that those should be of inferior quality, as there is no financial incentive to produce quality software. This ignores the fact that intrinsic incentives are in many cases more effective than monetary compensation [Gosh 06], and the fact that sometimes users are interested in improving the software that they are using [VH 03]. This second effect, called user-driven innovation, has been shown in past research to be a significant force. For example, around 25% of innovations in fields like software security, printed circuit boards CAD systems and library software were designed and introduced by advanced users. The same effect provides a fundamental design feedback,

	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 17 of 82
		Version: 1.0
		Date: Oct. 10 2007
		Status : Final Confid : Public

as large project collects both good and bad experiences in using the software (for example, the Ubuntu Linux “Testimonial and Experiences page³” that allows for a form of user-driven “steering” of the project and the identification of trouble points.

- **there is no support:** most large scale project are related to companies that provide paid-for support, in a way similar to that of proprietary software companies. The availability of source code and the modification rights gives also the additional advantage that support can be obtained even for projects that are no longer active, in stark difference with proprietary software where no code escrow clause was included in the acquisition contract.
- **FLOSS is inherently unreliable:** many believe that FLOSS, as developed in an open and unstructured way, is inherently of lesser quality when compared to proprietary software. The reality is that most FLOSS projects are organized in a semi-strict structure, and only very modular projects are inherently “bazaar-style”, allowing for large scale internal decoupling. In any case, the impact of FLOSS-style development has been assessed in several research papers, and for example in [Suc 04] we found: *“The hypothesis that open-source software fosters more creativity is supported by our analysis. The growing rate, or the number of functions added, was greater in the open-source projects than in the closed-source projects. This indicates that the open-source approach may be able to provide more features over time than by using the closed-source approach. Practitioners interested in capturing market share by providing additional features should look to the open-source methodology as a method to achieve this. In terms of defects, our analysis finds that the changing rate or the functions modified as a percentage of the total functions is higher in open-source projects than in closed-source projects. This supports the hypothesis that defects may be found and fixed more quickly in open-source projects than in closed-source projects and may be an added benefit for using the open-source development model.”* This is consistent with results from vendors of software defect identification tools, such as

³<http://ubuntuforums.org/forumdisplay.php?f=103>

Reasoning, that found that while the bug density ratio in initial project releases is on par with proprietary developments, it improves rapidly and for some projects defect densities are significantly lower than that of the average proprietary code [Reas 06a, Reas 06b]⁴. This was confirmed by other studies like the reports from Coverity.

The fact that FLOSS is overall reliable can be also inferred by surveys like [CIO 07], where 79% of respondents answered positively to the question *“My company's experience with open source products other than Linux has been so good we plan to expand their use”*.


In this sense, it should be no surprise that several FLOSS projects have received safety certifications, or have been used in medical devices, control systems and avionics. For example, the VISTA system is a large scale electronic health care system, developed by the US Department of Defense for its own veteran hospitals, and now used in more than 1000 hospitals and clinics in the US alone, along with many other installations across many countries. Other examples include the use of Linux in Siemens Magnetic Resonance Imaging systems used in diagnostics, the use of the open source ADACORE environment in in-flight avionics, the FIPS-140 certification of two of the most important encryption toolkits (OpenSSL and NSS), and many more.

If we take as an example the IEC 61508 safety integrity levels [Daf 06-2]:

SIL level	Dangerous failures/hour	Risk reduction factor
4	<10exp-8	>10000
3	<10exp-7	>1000
2	<10exp-6	>100
1	<10exp-5	>10

the UK Health and Safety Executive, in a study from 2002 [HSE 02]

⁴At a defect density of 0.09 defects per KLOC, the version of MySQL we inspected has a defect density that is about six times lower than the average of comparable proprietary projects.”

	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 19 of 82
		Version: 1.0
		Date: Oct. 10 2007
		Status : Final Confid : Public

found that Linux was robust enough, and that it could be certified up to SIL3 with limited effort. This would make it amenable for use in air traffic control displays, railways control systems and process plant control.

Myth #3: Big companies don't use FLOSS.


The easiest myth to dispel: apart from the large IT companies that are actively promoting open source software like IBM, HP, Sun, Oracle, and others, about 86% of Fortune 1000 companies are deploying or testing FLOSS, and a similar percentage is found in Europe [Aug 04]. Of those, 35% or more are deploying more than 20% of their systems as FLOSS, and 11% of companies report more than 20% of their applications as FLOSS. While usage in server-centric and IT infrastructure is more common, around 26% of large companies are mentioning the use of Linux on the desktop, and a much larger percentage are reporting the use of some other FLOSS packages, such as OpenOffice.org and Firefox on Windows desktops. A curious fact also evident from other surveys is that many companies and public administrations are not aware of their internal use of FLOSS, sometimes for simple ignorance of the licensing terms and sometimes because the product is offered or embedded in what seems like a traditional proprietary offering (for example, many security and networking products, or enterprise products like VMware ESX server, use FLOSS internally).

Myth #4: FLOSS is hostile to intellectual property.

There are several aspects that are referenced to this myth:

- **The GPL license is "viral":** the most widely used license does have a specific clause which mandates that when a software product that is derived from GPL software code is redistributed, the entire product must comply with the conditions of the GPL. This has prompted some companies to claim that *"the viral aspect of the G.P.L. poses a threat to the intellectual property of any organization making use of it"*⁵. The reality is that for most

⁵As mentioned by Craig Mundie, Microsoft's vice president, in a talk at New York University's Stern school of Business in 2001. Other representatives of Microsoft like Bill Gates said that "[the GPL] it makes it impossible for a commercial company to use any of that work or build on any of that work", and Steve Ballmer "Linux is a cancer that attaches itself in an intellectual property sense to

	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 20 of 82
		Version: 1.0
		Date: Oct. 10 2007
		Status : Final Confid : Public


scenarios, this clause simply provides a way to prevent appropriation of code without giving back contributions or credit, which is one of the reasons why many developers prefer the GPL to other licenses. Simple use of FLOSS in itself does not require any change to the license of internally developed software, and most companies routinely run proprietary software on top of GPL-licensed code like the Linux kernel.

- **The free software community steals the intellectual property of other companies:** this is mainly the byproduct of a legal case, in which the SCO company claimed in 2003 that IBM improperly included copyrighted material in the Linux kernel. In the original claim, it was mentioned that IBM *“put SCO’s confidential and proprietary information into Linux, the free operating system”*⁶ and that several millions of lines of code of the Linux kernel were stolen from SCO's Unix source code. Now, four years later, the judges have thrown out most of the claims, leaving less than 300 lines of code (mostly standard interface code) still under evaluation, out of more than 6 million lines of code of a modern Linux kernel. Recently Microsoft issued similar statements, with Microsoft's CEO Steve Ballmer⁷ claiming that *“that product (Linux) uses our patented intellectual property”*, and later numbering how many patents Linux and other FLOSS products were infringing Microsoft's intellectual property. The reality is that structured FLOSS projects do have strict policies for accepting patches and external contributions. As an example, the Eclipse project has a strict due diligence process, that covers external contributions, code rights assignments, code review and license compatibility. The Eclipse Foundation also uses automated tools to check for code copying, keyword scanning for words with legal significance and a controlled release review prior to updating the code [Cam 06]. Similar processes are in place in other FLOSS projects [Rig 06].

everything it touches ... if you use any open-source software, you have to make the rest of your software open source” (interview at Chicago Sun-Times, 2001).

⁶The transcript of the initial complaint and a full list of case documents (along with significant analysis) can be found in the GrokLaw site, at <http://www.groklaw.net>

⁷<http://blogs.zdnet.com/hardware/?p=154>

	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 21 of 82
		Version: 1.0 Date: Oct. 10 2007
		Status : Final Confid : Public

Myth #5: FLOSS is all about licenses.

While in a strict sense a FLOSS project is defined by its license, most aspects of open source are really related to the openness and collaborative aspects of the development, as described in chapter 1.


Myth #6: If I give away my software to the FLOSS community, thousands of developers will suddenly start working for me for nothing.

There is no guarantee that simply “dumping” source code on the web will make a FLOSS project appear, and there have been several examples of such behavior to be even negative (because the community may see this as “garbage dumping”). The reality is that for some collaboration to happen, there must be first of all a good communication, interaction strategy and effort in place. In addition, investing in community creation and dissemination efforts increase the probability of a bidirectional effort sharing. It is important to mention that surveys like OSSWatch or [CIO 07] found a significant proportion of companies and public administrations (between 14% and 25%) contribute back patches or participate actively in FLOSS communities.

Myth #7: FLOSS only matters to programmers, since most users never look under the hood anyway.

The fact that most users are not interested in the source code does not imply that having the source code available in itself is useless. Several positive aspects can be identified:

- The availability of the code allows end users to eventually pay someone for modifications or continuing maintenance even when the original FLOSS project disappears or becomes inactive.
- “Under the hood” there is not only code, but many non-code artifacts that are vital to a project, like translations, documentation, examples, etc. Many users can contribute in such aspects even if they are not programmers.
- For some projects, having the code available allows for a significant cost reduction or increases dramatically the flexibility

	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 22 of 82
		Version: 1.0
		Date: Oct. 10 2007
		Status : Final Confid : Public

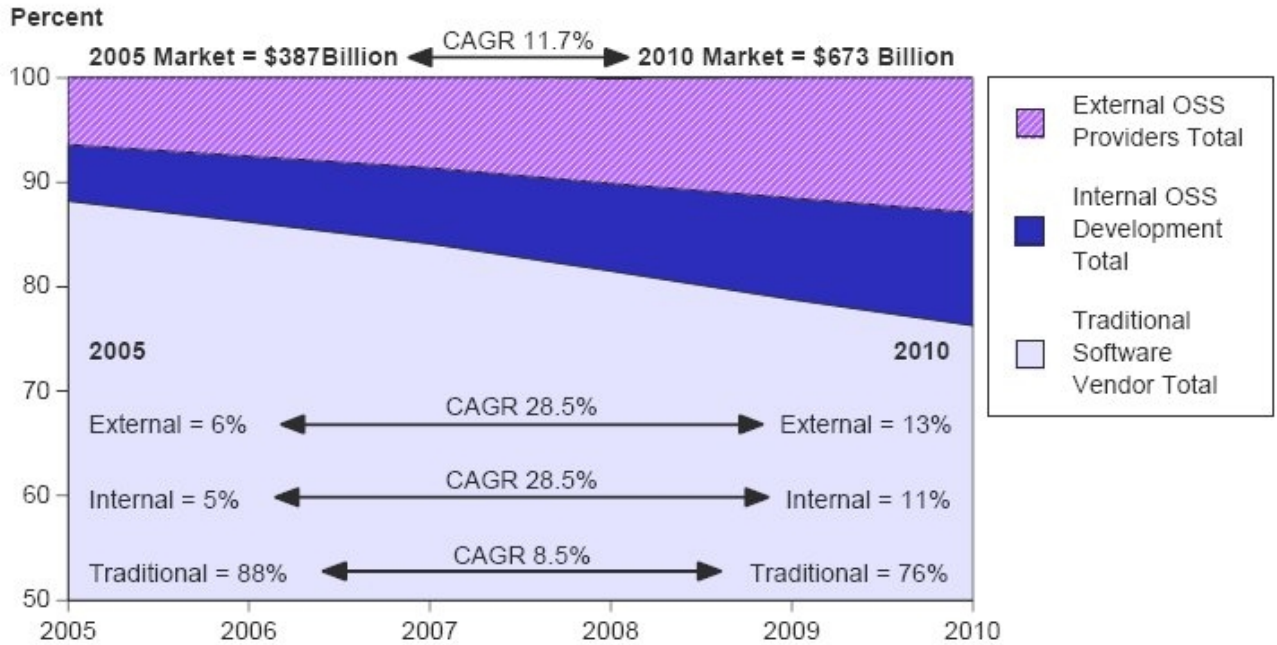
of the offered solution. For example, in a project called MuleSource (a sophisticated middleware system) it was found that 64% of users perform at least one source code modification.

Myth #8: There is no money to be made on FLOSS.

Even many researchers have proclaimed in a way or the other that the freely available nature of the code precludes any potential commercial exploitation. For example, in [Hahn 02]: *“The GPL effectively prevents profit-making firms from using any of the code since all derivative products must also be distributed under the GPL license”*. This of course collides with the economic results obtained by companies like HP (that in 2003 reported more than 2.5B\$ in Linux-related revenues), or the 400M\$ revenues reported in 2006 by RedHat. In [Gosh 06] it is evaluated that:

- Defined broadly, FLOSS-related services could reach a 32% share of all IT services by 2010, and the FLOSS-related share of the economy could reach 4% of European GDP by 2010.
- FLOSS directly supports the 29% share of software that is developed in-house in the EU (43% in the U.S.).
- FLOSS potentially saves industry over 36% in software R&D investment that can result in increased profits or be more usefully spent in further innovation.
- The notional value of Europe’s investment in FLOSS software today is Euro 22 billion (36 billion in the US) representing 20.5% of total software investment (20% in the US).

Similar measures are predicted by independent consulting groups like Gartner: in [Gar 06] it is predicted that two years from now, around 25% of the total software market will be FLOSS-based (either through external providers, or by internal developments).




Another relevant aspect is that since most companies adopting FLOSS report significant cost savings, these can be directly transferred to external professional services or incorporated as additional profit margin. For example, in [Inf 07]:

	1 or MORE Open Source Products Installed	1 – 25 Open Source Products Used	25-100 Open Source Products Used	More Than 100 Open Source Products Used
No savings	7%	8%	1%	5%
Too early to tell	44%	47%	27%	31%
10% - 20%	20%	19%	22%	14%
21% - 40%	12%	12%	15%	17%
41% - 60%	7%	5%	16%	10%
More than 60%	9%	7%	18%	24%
Costs went up	1%	1%	1%	0%

Q: Estimate the percentage of your IT budget saved each year by using open source projects Base: 1-100+ open source products installed (672), 1-25 installed (536), 25-100 installed (94), more than 100 installed (42).

In a survey of 800 IT managers, InfoWorld found that of all the FLOSS adopters, those collecting the most significant benefits are those that deploy more open source products, with 24% of the "large users" (more than 100 products) reporting savings of more than 60%. It is also interesting to notice that only a very small percentage (<9%) reports that there are no savings or that costs have increased compared to proprietary software.

	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 24 of 82
		Version: 1.0 Date: Oct. 10 2007
		Status : Final Confid : Public

Myth #9: The FLOSS movement isn't sustainable, since people will stop developing free software once they see others making lots of money from their efforts.

This is connected to the view of myth #2, the idea that FLOSS is developed by volunteers, and that companies can only profit in a parasitic way from the code that is developed for free. As discussed in that part, the reality is that in most projects companies and volunteers participate in a collaborative and non-competitive way; also, the most widely used license (the GPL) forced companies to reciprocate their efforts by making dissemination of the source code mandatory whenever there is dissemination of code derived from GPL projects.


Myth #10: FLOSS is playing catch-up to Microsoft and the commercial world.

The concept of software innovation is really rooted in two different aspects: technical innovation and field innovation. While technical innovation is mostly invisible to the user, “field innovation” (for example a new kind of application) is highly visible. Maybe because of this it is widespread the perception that most FLOSS software is sort of a copy of some other (desktop) oriented proprietary application.

The reality, on the contrary, is that most proprietary software is non-innovative in this aspect. While very few examples of new concepts (like Dan Bricklin's spreadsheet idea) can be found, most applications are matched to the tasks that people performs daily, and as such there is a strong disincentive to innovate. There are very few studies comparing FLOSS with proprietary software in a replicable and objective way, and one of those is [Kli 05]:

	New technology	New for a platform	Existing technology
New market	Radical invention (breakthrough) 5 (1.0%)		Marketing innovation 3 (0.6%)
Existing market	Technology modification 4 (0.8%)	Platform modification 52 (10.4%)	No innovation 436 (87.2%)

The end result is that from a field innovation point of view, around 12% of the projects in the sample are considered innovative, a percentage that is comparable to that of the proprietary software market. As for the technical innovativeness, the already cited [Suc 04] found that *“The hypothesis that open-source software fosters more creativity is supported by our analysis. The growing rate, or the number of functions added, was greater in the open-source projects than in the closed-source projects. This indicates that the open-source approach may be able to provide more features over time than by using the closed-source approach.”*

	Guide for SMEs Deliverable ID: D8.1.1	Page : 26 of 82
		Version: 1.0 Date: Oct. 10 2007
		Status : Final Confid : Public

3. Basic FLOSS adoption models

Within a company, the value that comes from FLOSS can derive from several different areas:

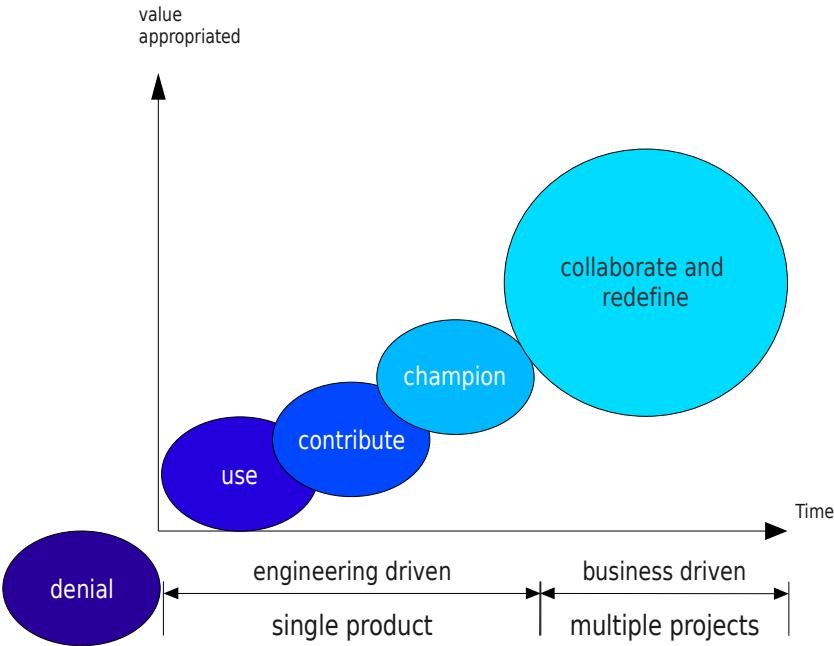
- basic substitution/migration: the use of FLOSS in the IT infrastructure, frequently in substitution of a proprietary software
- new deployment: the introduction of FLOSS for a new project internal to the company (adoption)
- selling services based on FLOSS
- selling products that contain FLOSS as a significant component

In this sense, a company may find useful FLOSS from a tactical point of view (FLOSS is cheaper to implement, with less constraint from a traditional vendor, or may help in introducing products in a reduced time to market) or a strategic point of view (creation of new markets, adoption of different business models). To be sustainable, a company must adopt a business model that provides a way to turn the FLOSS adoption into lower costs or increased revenues, and must also take


into account the fact that at least a part of the participant community may be out of control of the company (as it commonly happens in large scale FLOSS projects, most contributors are not working for a single company).

The FLOSS adoption ladder

These different areas corresponds to individual steps in the FLOSS adoption ladder, that can be summarized as (modified from [Car 07]):



In the first stage ("use") there is simple adoption or migration, usually without any additional contact with the community, of one or more FLOSS packages. This adoption is in many cases started in a grassroots way, directly by employees, and it is performed with the specific target of exploration or reduction of costs. Many examples of adopted packages in this area are related to desktop applications, like the Firefox web browser or the OpenOffice.org personal productivity application; in some cases, small single-purpose application servers

	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 28 of 82
		Version: 1.0
		Date: Oct. 10 2007
		Status : Final Confid : Public

are introduced, like mail servers or web servers for introducing web-based applications. At this stage, usually there is no or very little contribution back to the community, that in many cases is not even perceived as a peer in the potential interaction. However, most companies that started adoption of FLOSS for the internal IT infrastructure are actively extending it; for example, a [CIO 07] survey found that of those adopting Linux, 65% of companies are planning to extend its use, while only 1% plan for a use reduction. These positive results tend to increase familiarity with FLOSS in general and with the underlying collaborative model, and facilitate the upgrade to the successive steps.

In the second stage ("contribute") there is an active involvement by the company into the development of the adopted FLOSS project. This contribution may come directly in terms of code, or through participation in events, indirectly by sponsoring, or simply by acting as promoters of the project. This step requires an explicit support from management, and provides positive returns both for the project and for the company (that reduces the cost of having functionalities implemented, by sharing the development cost with the community); there is also an explicit recognition of the participation and activities of internal developers and their interaction with FLOSS projects. An example of company in this stage is Apple (as OSX leverages more than 340 different FLOSS projects).

In the third stage ("champion") the company is basing a significant part of the underlying business model on FLOSS projects, and as such devolves a significant effort in the participation activities. The basic support activities of the contribution stage is strengthened and extended, to make the company a key management point that manages not only internally-produced contributions, but external developers as well. This turns the company into a part of the much larger project ecosystem, and provides increased business opportunities thanks to this enlargement.


The fourth stage ("collaborate and redefine") is characterized by an extension of the cooperation model, from a disjoint collection of individual projects to a coordinated effort to influence the market and the customer's perception of the environment. Not only the company

changes most of its internal structure to accommodate open development practices, but also encourages the creation of a network of partners and independent projects, that are perceived as potential enlargements of the business ecosystem (even if some of those same projects can become potential competitors).

The cost and activities that are specific of each stage can be synthesized as:

Stage	Main cost centers
Use	Identification of potentially interesting software, adoption, migration, training
Contribute	development time, sponsorship
Champion	development time, sponsorship, community interaction, support to third parties
Redefine	development time, project and ecosystem coordination

It may surprise the fact that among the main cost centers of the first stage ("use") the identification of applicable software is prominent. This is confirmed by independent studies, like the EU COSPA migration project. Using data from [COS 05], we find that the "searching process" (that involves both searching for software and searching for documentation) is responsible for around 40% of the support costs, in some cases even surpassing the overall training costs of a large scale migration.

	Guide for SMEs Deliverable ID: D8.1.1	Page : 30 of 82
		Version: 1.0
		Date: Oct. 10 2007
		Status : Final Confid : Public


4. Finding and selecting software

As briefly mentioned in the previous chapter, the software selection process is an often overlooked but extremely important component of a migration or adoption of FLOSS. As mentioned in Appendix 1, there are more than 18000 mature and stable open source project, and most of these have no strict "promotional" budget or are not backed by companies that are able to provide marketing and dissemination support.

There are three separate steps that should be taken to successfully identify a set of FLOSS packages:

- identify your requirements
- search for packages matching your functional requirements
- select the appropriate package from the matching set

The first step is an often overlooked activity, but is crucial for a

	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 31 of 82
		Version: 1.0
		Date: Oct. 10 2007
		Status : Final Confid : Public

successful adoption; in many cases, there are no perfect matches for a given proprietary product, but equally good alternatives that perform the necessary activity as well (and sometimes even better). In this sense, a small shortlist of "required" and "useful" functions should be a first step in performing the selection.

After the shortlist, it is necessary to find the packages that may satisfy the given requirements. There are several important web sites that provide information on available software, both in an undifferentiated way (like SourceForge, that mainly acts as a project repository) and through detailed reviews and comparisons with proprietary software.

Forge-based sites:

these sites are mostly providing support and download services, and host a number of project that varies between 150000 (Sourceforge) to a few hundred; an integrated search functionality is provided. Most are based on SourceForge code, its reimplementations (GForge), or on collaborative development platforms that provide similar services (storage, email communication, code versioning and change support, bug tracking). Some of the most important sites:

<http://sourceforge.net/>
<http://savannah.gnu.org/>
<https://gna.org/>
<http://alioth.debian.org/>
<http://www.berlios.de/>
<http://codehaus.org/>

Software announce sites:

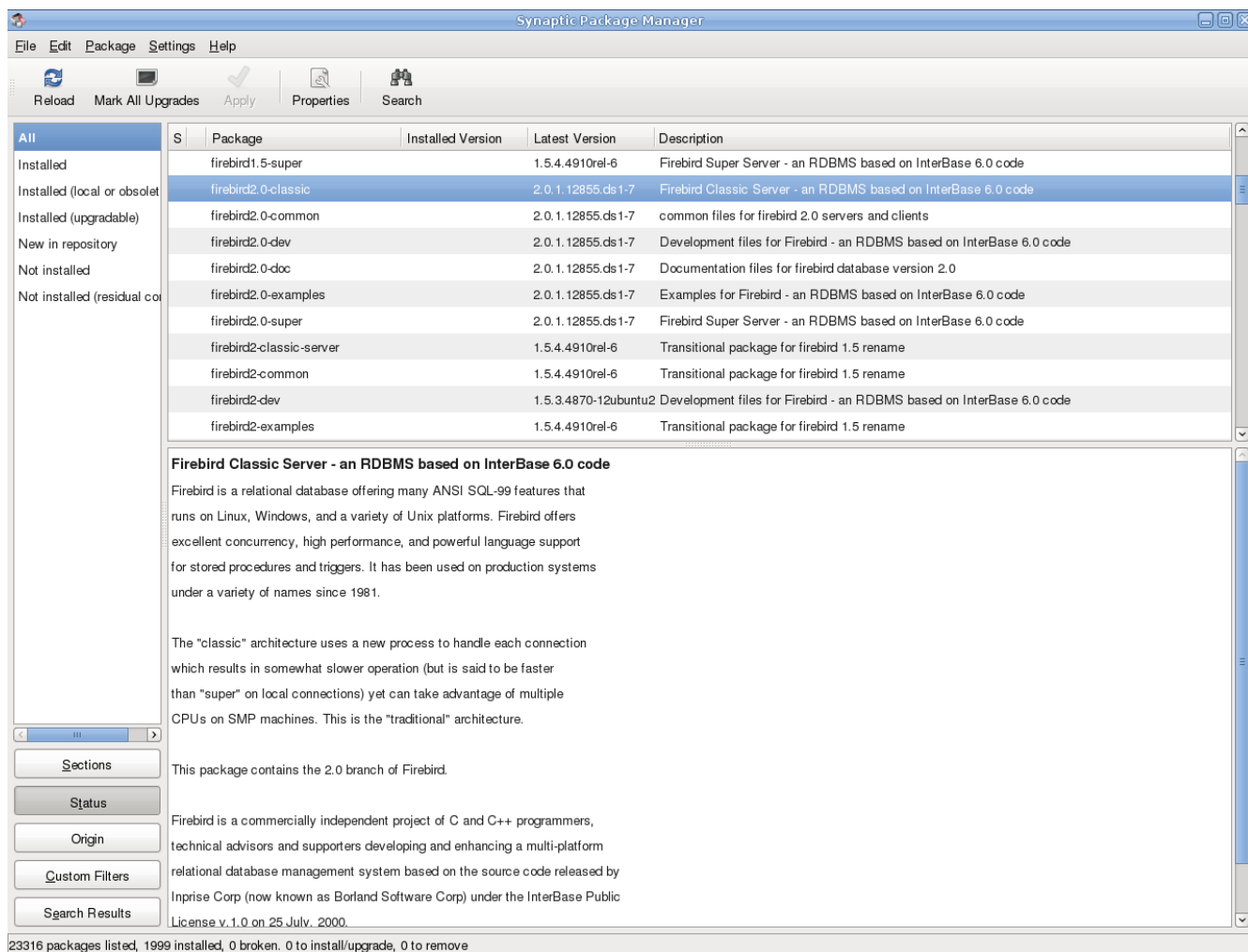
These sites are mainly news aggregators, that provide detailed information on recently announced versions of a FLOSS package, along with information on licenses, home page and screenshots.

<http://freshmeat.net/>
<http://sourcewell.berlios.de/>

List of software equivalents:

<http://www.linuxrsp.ru/win-lin-soft/table-eng.html>
<http://www.osalt.com/>

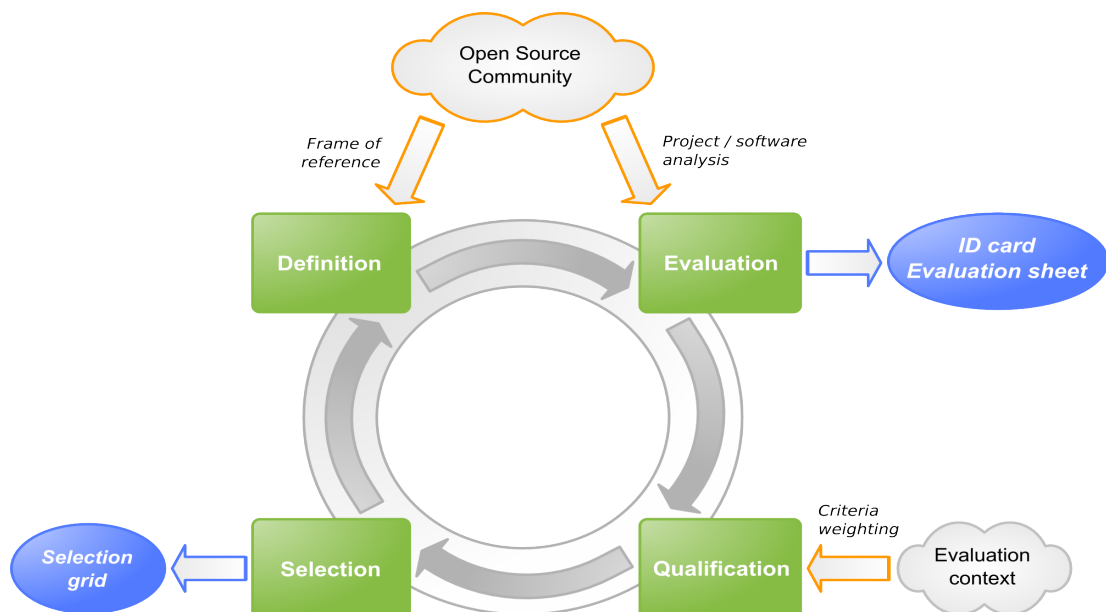
Most Linux distributions also include a package search tool, like Debian and Ubuntu's Synaptic tool:



This tool provides search and installation support for all the installable packages that are included in the distribution "repositories", specialized sites that provide binary packages of the available FLOSS projects. The repositories are divided usually into "stable" and "unstable" ones, to provide the end-users with the choice between stable software and the last version (with the latest features, but not as thoroughly tested). It should be noted that nowadays no modern, end-user targeted distribution require the user to see or


interact in any way with the FLOSS source code; in this sense, if to install a package it is necessary to perform code compilation or similar activities, the package itself should be considered experimental, and its adoption should be limited to where internal, specialized support is available.

Once a set of potentially useful applications have been found, it is fundamental to evaluate between the various applications. This can be done applying the QSOS methodology, created in the context of the EU project with the same name, and available at <http://www.qsos.org> . The project leverage previous activities in the same area, like the Open Source Maturity Model from Navica, OSMM from CapGemini or the Business Readiness Ratings; and uses a 4 step approach:



The methodology is divided into steps, with the "definition" step used to define the element used in the evaluation/selection/qualification ones. The definition is based on the following elements:

- **Software families:** hierarchical classification of software domains and description of functional grids associated with each domain
- **Types of licenses:** classification of free and open source licenses
- **Types of communities:** classification of community

	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 34 of 82
		Version: 1.0
		Date: Oct. 10 2007
		Status : Final Confid : Public

organizations existing around a free or open source software and in charge of its life-cycle.

The evaluation step is done in two steps, the first is the collection of the relevant and factual information on the FLOSS project (called in the QSOS terminology the "identity card") and the second is the creation of the evaluation sheet, based on three criteria:

- Functional coverage
- Risks from the user's perspective
- Risks from the service provider's perspective

The identity card collects the following data:

- **General information**
 - Name of the software
 - Reference, date of creation, date of release of the ID card
 - Author
 - Type of software
 - Brief description of the software
 - Licenses to which the software is subjected
 - Project's URI and demonstration site
 - Compatible operating systems
 - Fork's origin (if the software is a fork)
- **Existing services**
 - Documentation
 - Number of contractual support offers
 - Number of training offers
 - Number of consultancy offers
- **Functional and technical aspects**
 - Technologies of implementation
 - Technical prerequisites
 - Detailed functionalities
 - Roadmap
- **Synthesis**
 - General trend

- Comments

The evaluation sheet is based on a functional assessment, with a scoring rule that uses 0 to mark a functionality that is not covered by the FLOSS project, 1 for partial coverage and 2 for complete coverage (the product implements the required functionality). See Appendix 2 for a complete list of score tables.

After the individual evaluation, two different selection criteria can be applied: strict (direct elimination as soon as software does not fulfill the requirements formulated in the qualification step) or loose (rather than eliminating non-eligible software, it classifies them while measuring the gaps with applied filters).

The most relevant approach for SMEs is the loose selection, as the strict one may in several circumstances not return a suitable solution. The loose approach uses two weightings, one for functionality:

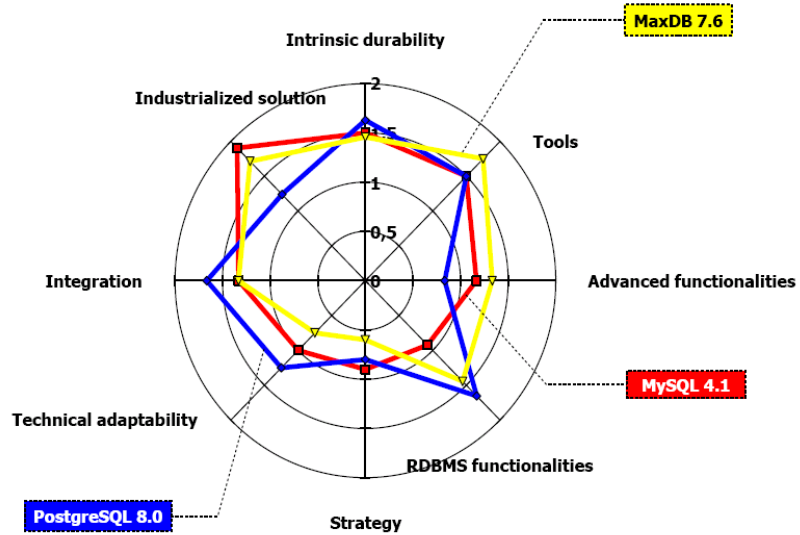
Level of requirement	weight
required functionality	+3
optional functionality	+1
not required functionality	0


and one for user's risk:


relevance	weight
-----------	--------

Irrelevant criterion	0
relevant criterion	+1 or -1
critical criterion	+3 or -3

another part of the QSOS project, the O3S tool allows for simple graphing and comparison:



	<p>Guide for SMEs</p> <p>Deliverable ID: D8.1.1</p>	<p>Page : 37 of 82</p> <hr/> <p>Version: 1.0 Date: Oct. 10 2007</p> <hr/> <p>Status : Final Confid : Public</p>
---	---	---


	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 38 of 82
		Version: 1.0 Date: Oct. 10 2007
		Status : Final Confid : Public

5. Best practices for FLOSS adoption

The migration and adoption process is a complex, multidisciplinary effort that touches different areas and require a complete understanding of how individual workflows are composed and executed and how people interacts with IT systems in their daily work. In this sense, a FLOSS migration is a major endeavor, and as most complex efforts can easily go wrong. There are several hurdles in the execution of a migration, and some of those hurdles can be avoided easily by using simple practices. Most of the difficulties are not really technical in nature, but organizational, and will require most effort from the upper management; another important aspect is the social impact of the migration (like user acceptance), that may require special attention.

Management guidelines

The main drive for a successful migration to FLOSS always starts with

	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 39 of 82
		Version: 1.0
		Date: Oct. 10 2007
		Status : Final Confid : Public

a clear assessment of the IT landscape, a clear vision of the needs and benefits of the transitions and continual support. The differences of OS development models and support may require a significant change in the way software and services are accounted for and procured, and in general a shift of responsibility from outside contractors to in-house personnel.


Be sure of management commitment to the transition

Management support and commitment have been repeatedly found to be one of the most important variable for the success of complex IT efforts, and FLOSS migrations are no exception. This commitment must be guaranteed for a time period sufficient to cover the complete migration; this means that in organizations where IT directors are frequently changed, or where management changes in fixed periods of times (for example, in public administrations where changes happens frequently) there must be a process in place to hand over management of the migration. The commitment should also extend to funding (as transitions and training will require resources, both monetary and in-house). The best way to insure continued coordination is to appoint a team with mixed experiences (management and technical) to provide continuous feedback and day-to-day management.

troubleshooting point: if the only people working on planning the migration is from IT/MIS, there may be insufficient information in upper management and financial planning for continuing the migration after the initial step.

Prepare a clear overview of what is expected from the migration or adoption, including measurable benchmarks

The transition can be started for several reasons, including better control on IT costs, independence from suppliers, flexibility or support of open data standards. To be sure that the migration is effectively producing benefits or is going accord to the migration plan, it is fundamental to know beforehand what indicators will be used to evaluate the progress. Those requirements must be realistic, in particular expectations of TCO reductions must be compared with publicly available data for comparison.

	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 40 of 82
		Version: 1.0
		Date: Oct. 10 2007
		Status : Final Confid : Public

troubleshooting point: if the only perceived advantage is that “the software comes from the net for free”, there may be a set of wrong assumptions that will probably lead to a final negative judgment on the migration.

Make sure that the timetable is realistic

The introduction of a new IT platform will always require a significant amount of time; as a rule of thumb the time to perform a full transition to FLOSS may be considered to be comparable to that of the introduction of a new company-wide ERP (enterprise resource planning application); for smaller transitions, time effort should be scaled accordingly.

Troubleshooting point: when migration time is measured in days, and no post-migration effort is planned, the process may be forced to a stop after the planned resources are expended.

Review the current software/IT procurement and development procedure

As implementation effort is shifted from commercial to open source software, the procurement and development process needs to be updated accordingly. In particular, the focus may change from acquisition to services, as less software is bought “shrink-wrapped” (commercially bought), and this change may require changes in how the internal IT budget is allocated.

Internally developed software will require a porting or a rolling transition to new software that is either multi-platform or accessible using standard interfaces (for example, web applications), and this should be taken into account in the overall IT plan.

Troubleshooting point: When no change of procurement and development is planned, the management may have not understood the scope of changed required for the adoption of FLOSS.

Seek out advice or search for information on similar transitions

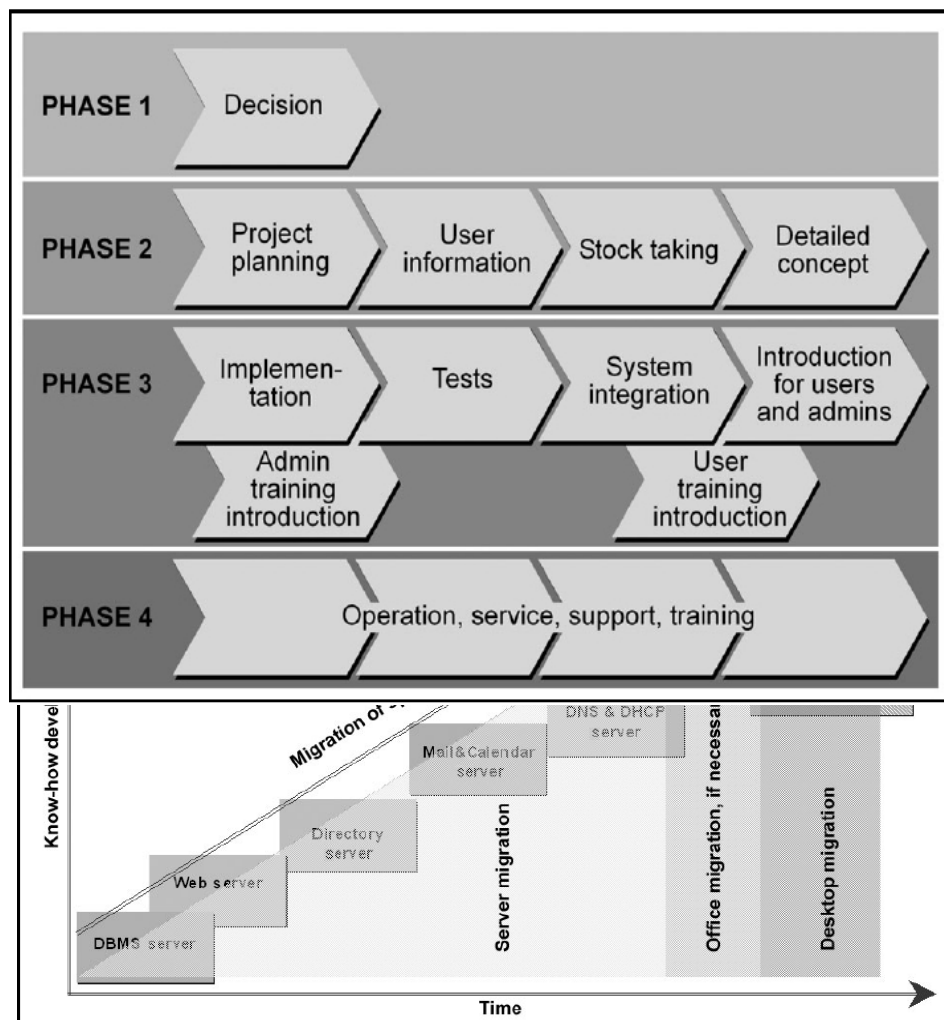
As the number of companies and administrations that have already performed a migration is now considerable, it is easy to find information on what to expect and how to proceed. In this sense, the


COSPA project has developed an online knowledge base that is accessible through the main COSPA site (www.cospa-project.org); public administrations can also contact their local Open Source Competence centre, that will provide information and support in the migration process.

Avoid “big switch” transition, and favor incremental migrations

Most large scale migrations that are performed in a single, large step (involving the abrupt change from one IT environment to the other) are usually marred by extremely high support and technical costs. While the need to support more than one environment does increase support and management cost, “gentle” or incremental migrations usually bring a better overall experience for the users and result in minimal disruption on business processes.

An example of gentle migration can begin with the migration of server side applications, that are usually standards or network-based and thus easier to replace, leaving desktop and user-facing applications last. Such a scheme can be depicted as: [KBST 06]



	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 42 of 82
		Version: 1.0
		Date: Oct. 10 2007
		Status : Final Confid : Public

Assign at least a person to interacting with the OSS community or the OSS vendor, and try to find online information sources

A significant advantage of OSS is the availability of online free resources, in the form of knowledge bases, mailing lists, wikis (collaborative sites) that may provide a substantial support in many cases comparable to commercial offerings. The biggest problem is the identification of such knowledge sources; in this sense assigning a resources to find, categorize and interact with such sources is a way to reduce the cost of support; a common way to provide a unified source of information is by setting up a small Intranet web page with links to online resources.

Troubleshooting point: when no one knows where to find information on the tools that are in use, or when everyone has to search on web sites on their own for finding usage tips.


Technical guidelines

A significant difference in FLOSS adoptions is the different development model adopted by most open source projects, and the difference in delivery of updates and support. This requires a change in the way adoption and updates are handled, to reduce as much as possible interoperability problems.

Understand the way OSS is developed

Most project are based on a cooperative development model, with a core set of developers providing most of the code (usually working for a commercial firm) and a large number of non-core contributors. This development model does provide a great code quality and a fast development cycle, but requires also a significant effort in tracking changes and updates. The adoption of an OSS package should be suggested when:

- when the project itself is “alive”, that is it does have an active development community. See the previous chapter on how to select and analyze a development project.
- when there is a clear distinction between “stable” and “unstable” software. In many projects, there are two distinct streams of development, one devoted to integrating the latest changes and addition, and another focused on improving stability and bug fixes; periodically, the developers will “freeze” development to

	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 43 of 82
		Version: 1.0
		Date: Oct. 10 2007
		Status : Final Confid : Public

turn the unstable version into the stable one, and create a new development, bleeding-edge version. This distinction allows the developers to satisfy both the users willing to experiment with the latest functionality, and those using the software for day-to-day operations, but requires an extra effort in collecting information and new versions.

If new functionalities or fixes are necessary, it may be easier to ask for a commercially supported version of the software; in many cases, the commercial vendor will also contribute financially to the open source project.


Troubleshooting point: when the IT manager or the developers think that OS is some kind of commercial software that someone has put for free on the net, and that it “just works”.

Create a complete survey of software and hardware that will be affected by the migration, and what functionality the company is looking for

There can be no successful migration when the initial situation is not known. Most companies and administrations have no process in place for auditing software and hardware platforms, and thus are unable to quantify the number of tools and software that needs to be replaced or integrated in an OSS migration. The survey process must also take into account the number of concurrent users, average use across the organization, and whether the software uses open or closed communication protocols and data formats. This survey will be the basis for the decision of what users will be migrated first, and for taking into account the cost of software re-development or migration to a different data format. Automated software inventory tools are readily available, and may reduce the cost of performing the inventory and allow for a stricter control on installed software (thus reducing the maintenance cost).

Some of the aspects that should be surveyed are:

- used data format, both at the document exchange level, database and network protocol level
- list of used applications, including those internally developed, macros and active documents
- available functionality
- shortcomings and problems of the current infrastructure

	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 44 of 82
		Version: 1.0 Date: Oct. 10 2007
		Status : Final Confid : Public

It is fundamental that the migrated software can fulfill the same functional requirements of the current IT infrastructure, and usually improve on that in functional terms or in inherent quality (like availability, reliability, performance).

Use the flexibility of OSS to create local adaptations


The differentiating thing of OSS is the flexibility and freedom that it gives to users and developers in creating new versions or adapted versions of any package. This flexibility can greatly enhance the perceived value of OSS, for example it is possible to create customized packages that contain local configurations, special fonts and other supplemental material like preset macros and templates commonly used in the company. Also, custom look and feel may significantly improve user acceptance, both by presenting a nicer looking desktop, and by maintaining common links and menu entries.

These customization can be integrated in a simple way in the most used Linux distributions, or by creating a local repository of software. Note that in many cases, it is not necessary to produce software or code, as most adaptations are related to selecting the appropriate package, change the graphical appearance, or providing templates and presets.

There is much more software available than what is installed by default

Licensing or design issues limit substantially the amount of software that is usually included in the default install of the most used Linux distributions. For example, only a few include playback capability for the most commons audio and video format, due to licensing and patent issues; for the same reasons, some packages that may be of interest to only a minority of users are not included.

For this reason, it is important to research and include in the default installs additional package that may help in the transition period; such packages include additional fonts, multimedia tools, and other packages that may be useful in a mixed environment.

	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 45 of 82
		Version: 1.0
		Date: Oct. 10 2007
		Status : Final Confid : Public

In selecting packages, always favor stability over functionality

Among the many potential packages available for every function, there is always a balance between functionality and stability. In general, among the potential candidate packages that satisfy the functional requirements for the migration the preference should be given to the one that is more stable, thus having a longer real-world usage (and thus more information available for the administrator) and lower variability between different releases.

Troubleshooting point: When the IT administrator wants the latest version of everything on user's desktop.

Design the workflow support infrastructure to reduce the number of “impedance mismatches”


Every transition from an ICT infrastructure to another leads to some “impedance mismatch”, that is to small differences and incompatibilities; this can be observed for example by translating documents from one data format to another. The overall infrastructure should reduce the number of such transition points, for example by redesigning the document templates in the ODT open format instead of reusing previously developed versions made using proprietary tools. This reduces greatly the formatting and style differences that arise when one format is translated into another.

Introduce a trouble ticket system

A difficulty of every new IT deployment is the assessment of user satisfaction and the degree of acceptance of the new solution, especially in medium sized companies when user feedback is more difficult to collect. An online trouble ticket system may provide an easy and simple way to collect weak points in the deployment, and can help in identify users that may need additional training by analyzing the per-user submission statistics. It may also point to weaknesses in the deployment, for example by pointing to several trouble tickets related to a specific area.

Compile and update a detailed migration workbook

A large scale migration effort requires a coordinated action, and clear

	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 46 of 82
		Version: 1.0 Date: Oct. 10 2007
		Status : Final Confid : Public

and updated information. The best way to provide this information is through a “migration workbook”, a single information point that provides the collection of documentation prepared for the migration (including the rationale, the detailed plan and the technical documentation) and the timetable, updated according to the project progress. This also simplifies project management when there is a change in the team performing the migration.

Social guidelines

A migration or adoption is not only based on technical basis, but does also have a social impact. As FLOSS is still not widely known, many myths and preconceived ideas may be hampering adoption by end users.

Provide background information on OSS

A significant obstacle of OSS adoption is the acceptance by the user, that usually has a very limited knowledge of open source and open data standards. In many cases, OSS is perceived as lower quality as it is “free”, downloadable from the internet like many shareware packages or like amateur projects. It is important to cancel this perception, and to provide information on how OSS is developed and what is the rationale and business model that underlie it. The chapter on "FLOSS myths" may be a starting point for providing factual information.

Don't force the change on the users, provide explanations

The change of IT infrastructure will force a significant change in how the users work and use internal resources; this change may cause resistance by the users. Such change may be simplified by explaining clearly why and how the change will happen, and what benefits will be introduced in the long term both internally (like lower cost, better flexibility and security) and externally (openness, adherence to international standards, less burden on external users). It is important to provide enough information and support to be able to skip the “opposition gulf”: [IBM 06]



Troubleshooting point: when internal users believe that the migration is done to software less

Use the migration as an occasion to improve users skill


As training for the new infrastructure is required, it may be used as a way to improve overall ICT skills; in many companies and public administrations for example little formal training is usually performed on users. This helps not only in increasing confidence, but can also used to harmonize skills among groups and in general improve performance.

This may rise some resistance from the so called “local gurus”, that may perceive this overall improvement as a lessening of their social role as technical leaders. The best way to counter such resistance is to identify those users, and suggest them to access higher-level training material (that may be placed in a publicly accessible web site, for example).


Also, it may be useful to identify local “champions”, that is local FLOSS enthusiasts, that can provide peer support to other users, and offer them additional training occasions or management recognition. In general, it is useful to create an internal Intranet accessible page that provides links to all the different training packages.

Make it easy to experiment and learn

The licensing freedom that is the main point of OSS allows for free redistribution of software and training material; in this sense, providing users with Linux live-CDs (that require no hard disk

	Guide for SMEs Deliverable ID: D8.1.1	Page : 48 of 82
		Version: 1.0 Date: Oct. 10 2007
		Status : Final Confid : Public


installation) or printed material that can be brought home may help in overall acceptance.

	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 49 of 82
		Version: 1.0
		Date: Oct. 10 2007
		Status : Final Confid : Public

6. FLOSS-based business models

One of the first categorization of potential business models was designed in 2001 in the work of the European Working Group on Libre software[DB 00]. The taxonomy, adapted to the recent developments of the market, is:

- Externally funded ventures
 - Public funding
 - `Needed improvement' funding
 - Indirect funding
- Internally funded or revenue based
 - `Best knowledge here" without constraints
 - `Best knowledge here' with constraints
 - `Best code here' without constraints
 - `Best code here' with constraints
 - `Special' licenses
- Unfunded developments

	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 50 of 82
		Version: 1.0 Date: Oct. 10 2007
		Status : Final Confid : Public

Externally funded ventures

We consider in this category groups or companies which develop open source software through the initiative (at least in the financial sense) of some external organization. Usually those external entities determine how the funds are to be spent, and where the development efforts are headed. The developer entity just follows those more or less strict guidelines. In some sense, it could be said that the external entity ‘sponsors’ the development of some given open source software. In this category, we can distinguish at least three models, according to who funds the project and why. We have called them public funding, ‘needed improvement’ funding, and indirect funding.

Public funding

Working groups or individuals receive funding for the development of a good software product, documentation, test cases or whatever. Usually, the only constraints imposed by the funding entity are that funds must be used to complete the project. This is typical of large computer science projects, and the funding usually comes from universities or from national science grants. In fact, many large projects in radioastronomy, computational chemistry, and biology are funded this way. In addition, some consortium for the development of Internet tools and technologies have (or have had) such a funding structure. It is important to notice that in these cases the funding institution is not expecting to recover the investment, or to directly benefit from it. Usually, some expectation of social improvement is the reason for the funding.

"Needed improvement" funding

A company or organization may need a new or improved version of a software package, and fund some consultant or software manufacturer to do the work. Later on, the resulting software is redistributed as open source to take advantage of the large pool of skilled developers who can debug and improve it.

A good example of the advantages of this model can be found in an article written by Aari Jaaksi, open source manager at Nokia, describing the experience of designing the Nokia N770 and N800


products, based on Linux: "The biggest cost savings came from the utilization of already available components. We utilized several free components and subsystems as such, with no modifications. We also improved several components to better meet our requirements. Such improvement is cheaper than creating the needed functionality from scratch. Some two-thirds of the code of the Nokia 770 is licensed under an open source license. These components made it possible for us to build the software cheaper than we could have done using closed and proprietary technologies" [Jaak 06]

In [Gosh 06] it is estimated that it is possible to obtain savings in terms of software research and development of 36% through the use of FLOSS; this is, in itself, the largest actual "market" for FLOSS, as demonstrated by the fact that the majority of developers are using at least some open source software within their own code (56.2%, as reported in [ED 05]).

In at least one instance the benefits of using FLOSS for product development have been evaluated, in the context of the European INES project[INES 06]. The project researched the use of FLOSS within industrial control systems developed by European SMEs, and measured the resulting economic impact:

Economic benefit	% of companies:
Internal Replications	100%
Increased Profit	100%
Reduced Time to Market	84%
Reduced Development Costs	79%
Reduced Product Costs	79%
Improved Code quality	79%
Improved Design Re-use	79%
ROI > 200 over 3 years	74%
Increased Product Reliability	68%

It is interesting to observe that companies that are adopting this model in many case contribute back the code that is developed even when not explicitly forced by the FLOSS project license, to reduce the cost of integrating product-specific patches and to leverage external support.

	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 52 of 82
		Version: 1.0
		Date: Oct. 10 2007
		Status : Final Confid : Public

Indirect funding / Loss-leader


A company may decide to fund open source software projects if those projects can create a significant revenue source for related products, not directly connected with source code or software. One of the most common cases is the writing of software needed to run hardware, for instance, operating system drivers for specific hardware. In fact, many hardware manufacturers are already distributing gratis software drivers. Some of them are already distributing some of their drivers (specially those for the Linux kernel) as open source software.

The loss-leader is a traditional commercial model, common also outside of the world of software; in this model, effort is invested in an open source project to create or extend another market under different conditions. For example, hardware vendors invest in the development of software drivers for open source operating systems (like Linux) to extend the market of the hardware itself. Other examples are related to the establishment of a platform or a specific protocol; for example the Eclipse project was extremely successful in creating a large ecosystem of tools and projects that complement and enhance it. Most companies have dropped their own internally-developed integrated development environment, and are using Eclipse as a basis even for commercial products.

Internal use

Some projects can get started as a lower-cost alternative to proprietary systems. In this case, the developer company does not have (at least in the beginning) any plan to get external income related to the sale of the software or services related to it. The company develops some system because it is useful for them, and later decides to make it open source, and distribute it widely, just to benefit from the open source development source. Probably they will get some contributions, improvements and bug fixes from external developers interested in the software, and some bug reports. Later on, the product may even reach some market acceptance, and the developer company could even get some economic benefits from it.

For instance, a large enterprise with several thousand desktop computers can decide to create some software internally, and make this software available under an open source license to get the

	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 53 of 82
		Version: 1.0
		Date: Oct. 10 2007
		Status : Final Confid : Public

benefits of a larger base of developers that may be interested in helping out. An interesting aspect is that recent surveys found that 25% of companies are working with other companies in the same sector to develop industry-specific open source software [CIO 07].


"Best knowledge here" without constraints

In this model, a company works as a paid consultant, with contracts granted on the basis of the higher level of knowledge of their employees. Any company can implement this model, as there are no limitations that prevent a competent technician from gaining an arbitrarily deep experience of open source software systems. Of course, this also means that any firm using this model is exposed to the risk of being superseded by someone else, if the level of competence is reached but not maintained. This is one of the pure "service based" models, that will be further refined later on in this chapter.

"Best knowledge here" with constraints

To prevent competitors from "stealing" customers, a firm can place arbitrary limitations on the process of knowledge sharing, through patents or through additional copyrights that are not conferred in a direct way through the FLOSS license. It can be implemented by placing under a more restrictive license just a small (but fundamental) part of the code, usually considering it as a "black box", or by adding a set of copyrighted materials not freely redistributable, and adding in the license an obligation to show them to the end-user ("badgeware"), thus preventing others from appropriating the code.

As a special case, there may be a need for external, non-code related conditions (like code certifications) that can be inherently costly to reproduce, and those can be added to a code distribution to create a non-transferable asset. For example, the CODE*ASTER project is a complex simulation systems used by the French utility company EDF in systems as complex as nuclear power plants. The project has a GPL version, and a quality-checked and certified version that has passed the national certification tests for use in safety-critical systems design. Other examples are security certifications like EAL4+ that have been recently obtained by major Linux vendors.

	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 54 of 82
		Version: 1.0 Date: Oct. 10 2007
		Status : Final Confid : Public

"Best code here" without constraints

In this model, a company develops some open source code, and sells consulting and maintenance services on it. This is similar to "best knowledge here", but with an additional advantage in terms of time, since a competitor needs some months to create a similar code, or to understand all the intricacies of someone else's source. This gives a time advantage to the company or group that creates the software in the first place.

"Best code here" with constraints/Time-decaying licenses


An interesting twist in licensing for OSS is that of time-decaying licenses, where a software artifact changes license with time or with some specific event (for example, the release of a new version of the code). The first known example of this model was the Alladin Ghostscript postscript interpreter, and recently some security companies provide up-to-date security signatures to paying customers, and release them under a public license after some days.

This model is especially suited to rapidly changing software or other material (for example, security and virus signatures) and less practicable for software, because the old version becomes a basis to create an improved product that may be competitive with the one under the commercial license. This is exactly what happened to Alladin, that found an open source competitor (GNU Ghostscript) based on a previous version of the code, plus many improvements contributed by open source developers.

Dual licensing

One of the few models that have no counterpart in the commercial software world, Dual licensing is used by companies that want to profit from the companies that want to use or leverage an open source package without standing the redistribution conditions of the OS license. For example, the MySQL database has two licenses, one GPL (for OSS usage) and a commercial one. The customer that wants to use MySQL in a commercial product without distributing the code pays for a commercial license.

Many other project are starting to use such a scheme, that mixes the

	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 55 of 82
		Version: 1.0
		Date: Oct. 10 2007
		Status : Final Confid : Public

traditional commercial software model and allow to pay for continued development; the downside is that the model can be effectively used only for source packages that needs to be linked in with the code for maximum efficiency or because there is no common protocol for data exchange. This means that for example dual licensing is difficult for packages like mail servers, that use common standardized protocols to communicate with mail clients.


Dual licensing requires some specific legal and community aspects to be handled; for example, patches or modifications from external contributors require an explicit author acknowledgment of both licenses, and management of the community requires an accurate management of the border between the commercial and open source aspects of the project.

Unfunded developments

If there is enough 'network effect', there may be no need for funding, just a minimal effort for the organization of releases and patches. Examples of these kinds of open source projects are the Linux kernel, GNU/Linux distributions like Debian, BSD-based operating systems such as FreeBSD, NetBSD, or OpenBSD, and the Mesa OpenGL-like library. These efforts started in many cases as the effort of a single man, or of a small group, and through good organization and volunteer work they created an extended networked structure that maintains the code. Even with some (limited) funding for some projects, all of these efforts become successful without an external grant or without explicit money offerings. In fact, this is the case for hundreds of small open source projects.

Specialized Service-based business models

Service-based business models are based on the idea of optimization, that is the capability by a specialized company to provide a service at an overall price for the customer that is less than the one the company would incur in if doing it by themselves. To get an overview of the areas that are subject to this potential optimization, we can provide an overview of the steps that are part of the adoption of a new ICT technology:

	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 56 of 82
		Version: 1.0
		Date: Oct. 10 2007
		Status : Final Confid : Public

- software selection (if Off-the-shelf components are used)⁸
- installation
- integration
- technical suitability certification
- legal certification
- training
- ongoing maintenance and support contracts⁹
- at the end, migration from old system to the new one

As we will see, only a few companies are specialized in a single model, but compose them together to create service packages, in a sense making these sort of "business models building blocks"; we will however first provide an overview of each block to provide an estimate of effort and difficulties that are inherent in each step.


Software selection support

Software selection is really a multi-step phase, that starting from the identified needs and a knowledge of the software market selects a combination of packages that minimize the amount of code that needs to be developed. This minimization process is complex, taking into account not only the technical characteristics of the software being considered, but also must provide an evaluation of the "liveliness" of the OS project, the probability that its development will continue, and the availability of consultants and documentation.

A company that wants to offer this kind of service needs a substantial investment in terms of knowledge of the different packages and tools available. As the number of projects that are amenable of business or PA use can be estimated to be over 18000, there is a substantial effort just in following the new updates or announcements. This is further complicated by the fact that most project do not have an explicit "marketing mechanism", that spreads information on features and capabilities on a software package like commercial software firms. This means that companies that want to offer software selection

⁸ there is an additional first step, identification of needs, that is not in itself specific to OSS, but is usually part of the responsibilities of the internal ICT staff of the company or the administration that needs to perform the migration. For this reason it is not included in the list

⁹ Keen, P. "managing the economics of information capital": maintenance is 40% per year for 5 years on average of the initial cost of software

	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 57 of 82
		Version: 1.0 Date: Oct. 10 2007
		Status : Final Confid : Public

consulting services must dedicate a certain effort just in monitoring web sites and mailing lists, and extract from there information on new versions or new packages; this effort can be estimated in 1 man/hour per day for limited segments (for example, only Java enterprise middleware) to 5 man/hours per day for many different software segments.

The actual consulting activity is fairly simple, and consists in in-depth interviews and analysis of the need of the customer, followed by the preparation of a list of suggested packages. It is also possible to estimate the cost of integration, using data from the software engineering community related to COTS projects (Common Off The Shelf).


Installation support

A very common support activity in the OSS community is that of installation; this comes from two different aspects: the great modularity of software (that forces the installation of many, different components to create a working system) and the relative unavailability of sophisticated software installers, common in the commercial world.

This is however changing, thanks to the standardization in the Linux world of packaging systems, that makes nearly non-necessary the installation of software from source code components. The availability of package installers based on the RPM (RedHat package Manager, used also by Novell's Suse Linux and Mandriva Linux) and DEB (used by Debian and derivatives, like Ubuntu) augmented with dependency maintenance systems have greatly reduced the complexity of installation, now mostly related to the modification of the suitable configuration files to adapt the installation to a specific ICT environment.

Integration support

Another of the most common steps in OSS-based consulting is the integration step, that relates both to the specific configuration step necessary to “fit” an open source component in an existing structure, and to the custom development necessary to add the missing

	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 58 of 82
		Version: 1.0
		Date: Oct. 10 2007
		Status : Final Confid : Public

functionalities or correcting the incompatibilities.

Integration may require a substantial effort for large scale projects, with a relatively large amount of custom coding or the integration of commercial components if no other choice is possible. This variability is the reason behind the strong push towards standards (both de-facto and de-jure) that is the simplest way to reduce interfacing cost between disparate software components.

Technical suitability certification

This is mostly done by integrators and external consultants, and may come in two shape: certification of adherence to an international standard (for example security or quality standards) and certification of suitability for a specific environments. In a sense, in both cases the integrator or certifier provides an insurance that the software package complies with a specified set of rules, and is legally liable for such compliance. Limited scope certifications, like security assurances, are quite within scope of SMEs, while large scale quality assurance of components is quite difficult to attain if the open source project itself does not have an in-place explicit mechanism for project management¹⁰.


Most Linux distributors performs this suitability test in a very simple way, by selecting the most plausible candidate version of a source code package depending on the distribution target (for example, in so called "enterprise edition" distribution only stable versions are used, while for "bleeding edge" distributions the latest unstable version is selected).

Legal certification

This is a relatively recent model, that emerged from the perceived problems of mixing code from multiple licenses, and from several lawsuits¹¹. Legal certification is related to the following areas:

¹⁰For example, the open source CODE-ASTER simulation package by EDF (the French utility) is quality certified and also certified suitable for use in the simulation and design of nuclear power plants; and the AdaCore ADA environment (based on open source components) is certified for avionics and high-availability environments.

¹¹It is interesting to notice that most of these lawsuits are only marginally related to open source licenses, and that the uncertainty has been in some way spread by commercial companies that are being

	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 59 of 82
		Version: 1.0 Date: Oct. 10 2007
		Status : Final Confid : Public

- correct use of OSS and commercial licenses
- patent certification
- other Intellectual Property certification

The first area is related to the mixing and correct use of components, that may have different licenses and different restrictions. While more than 70% of the open source code is actually released under the GPL, more than 50 other licenses exist, and some fundamental components are released under a non-GPL license (the Apache foundation software, Mozilla/Firefox or the Eclipse integrated development environment).

When using and integrating many different components, it is fundamental to be able to verify that all code is properly used and accounted for. This is really a task that requires legal capabilities, more than technical ones, and for this reason is perceived by the OSS community to be a “tangential” model.


Patent and IP certification provides a form of “insurance” against third party claims on software patents or other copyrighted material that may be in the OSS used in a project; as any insurance form, it is quite demanding in terms of monetary funds, as patent claims may give raise to multi-million Euro lawsuits (see for example the recent patent lawsuit by Eolas against Microsoft corp. with more than 500 million dollar in requested damages).

Training

Training is another commonly found business model, as many open source projects do not have an official, sanctioned training process that is comparable to that of commercial companies. Training is usually personnel-intensive, and requires some effort for the creation of the initial training material to be used during the courses. A good estimate of work needed is that it is necessary to invest around 3 to 8 hours of course material preparation for each hour of training delivered¹².

threatened by open source in their market.

¹²The variability depends on the complexity of the course and the specificity of the knowledge to be transferred.

	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 60 of 82
		Version: 1.0
		Date: Oct. 10 2007
		Status : Final Confid : Public


The simplicity of the model and the fact that it does not require software development means that it is quite easy for established training companies to compete for offering such services; also, the largest OSS project also usually have an official training programs (for example JBoss, Linux distributors RedHat and Novell/Suse).

Ongoing maintenance and support contracts

For most complex systems there is a continuous need for support and maintenance, both for bugs and feature enhancements and for the adaptation of the system to the changing IT environment. Support contracts usually are time-based (the most common is a contractual period of one year, renewable) and level-based. Levels are commonly three (corresponding to “bronze”, “silver” and “gold” support services), with varying degree of guaranteed service. For example, bronze level usually provides email-based support during work hours and access to a knowledge base; silver adds voice support and precedence of incidents over bronze contracts, and gold adds 24/7 live support¹³. While it is reasonably easy for an SME to offer standard support services, 24/7 offerings may require a slightly larger personnel base to guarantee coverage under every circumstance. Another model that is gaining ground is the acquisition of “tokens”, that are later used to buy specific support activities (for example, a support request may require one token, and an urgent one-“priority” may be bought for three tokens). This way, users may decide in a flexible way how to leverage the support offer without restrictions.

Taking into account the characteristics of support questions, it is possible to observe that most calls are easily answerable, even with only moderately skilled people (around 80% are “easy” calls); the remaining 20% usually require a much greater effort. It is possible sometimes to create “pyramids” of support, where one company provides support for those 80% of easy calls, and moves the harder ones to another company that is more specialized on a specific package or a specific issue. This requires of course the capability of categorizing calls appropriately, and requires the existence of specific support contracts between the participants; this is usually possible

¹³This subdivision has been extracted from support contracts of several ICT support vendors, but slight variation may be found- for example, 4 levels instead of 3, or different kind of support material other than the knowledge base.

	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 61 of 82
		Version: 1.0 Date: Oct. 10 2007
		Status : Final Confid : Public

only if the customer base is large enough, and so is more amenable to the medium companies.

The support model is used by many companies that turned a commercial package (not completely successful in the commercial market, or unable to completely fulfill its market potential) into an open source one; the underlying idea is that the authors of the code are supposed to be the most qualified experts for support it. The first famous example of this model was the Zope application server, with many others in active existence (for example, the computer aided design OpenCascade toolkit, Compiere, Alfresco and many others). It is interesting to notice that contribution from the outside are usually received from outside participants even in the case of very specific application areas, like for OpenCascade¹⁴


Migration services

Similar to integration services, migration is based on the deep knowledge of both the starting and end IT environment. Most migration services are based on software packages that help in automating the migration (for example of user configurations), or on pre-configured “packages” of OSS that provides complete substitutes of proprietary environments. Examples may be mail/groupware systems or desktop operating system replacements. Migration services usually require a specific integration step in addition to the base migration, and for some large scale effort may require coordination among different companies, offering coordinated service (for example, one specialized in porting custom code, one in migrating mail services, etc.)

Commercial-on-open

One of the simplest model for software companies is selling a proprietary software package on an open source one. It may be simply a matter of running platform (like having a commercial package running on Linux) or it may leverage an open source project with some commercial module. Examples of this abound, from commercial database systems, proprietary payroll or financial applications, to

¹⁴It has been reported that 20% of the “package value” of OpenCascade has been contributed by outside partners and developers; both in term of code and documentation and ancillary material. This percentage has been found in other projects, like JBoss.

	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 62 of 82
		Version: 1.0
		Date: Oct. 10 2007
		Status : Final Confid : Public

module designed to improve the usability and manageability of an existing open source project.


This second example is becoming one of the main options for funding an OSS project, and leverages the development of the OSS component to provide added value that may be of interest only to a part of the community, for example providing easy-to-use interfaces for a complex system. As the example in the previous section on time-decaying licenses, if the project is successful there is a risk of competition with an open source project designed to fill exactly the same need.

Any company that plans to follow this model should devote some effort to track the evolution of the OSS platform, and somehow participate (for example, with an active participant in the mailing lists of the project). This has the double advantage to provide an insight into the evolution of the platforms and new, potentially useful features, but also to be “good citizen” of the OSS project.

Mediation services

Mediation services are relatively new on the market of OSS models, and are based on the fact that for companies it is difficult to interact with sparse communities like some OSS projects. Mediation services provide a sort of a single point of contact, that gathers information from the developers, mailing lists, forum and such and forwards requests and bug-fixes back. These services are especially useful when the company is willing to pay for modification or changes to the code, but is unable to find a suitable service company. Usually these mediation companies try to contact directly the developers, or to find support companies that demonstrate experience in the specific package; after development, they add some certification and integration effort to deliver a single package to the customer.

This is useful especially for large scale efforts, where many different communities may be involved, or when there is no clear choice to ask for support or development. Large scale projects (like Apache, JBoss and others) usually have one or more company that provides already this kind of mediation.

	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 63 of 82
		Version: 1.0 Date: Oct. 10 2007
		Status : Final Confid : Public

Custom development

Another model that is just an application of a traditional one, the custom development is simply the offering of custom coding on an open source project. As such, there is usually a form of specialization on a single project or class of projects (for example, device driver development or open source-based J2EE systems). A company that wants to use this model should add to the traditional model an activity related to tracking the evolution and roadmap of the project on which it is specializing, in a way similar to that described in the previous commercial-on-open model.

Assessment of FLOSS business models usage


To assess the real business models adopted by FLOSS companies, we prepared an initial list of 120 companies using some popular open source news websites as source¹⁵; this list was further refined by eliminating companies that were not really adopting FLOSS, even using a very relaxed definition. In the specific, any company that allowed source code access only to non-commercial users, or that did not allowed for redistribution was dropped from the list; also, companies for which no information was available, or for which no clear product or service was identifiable was equally eliminated.

One of the companies included (Sourceforge, from the OSTG group) is not open source in itself¹⁶, but represents an example of an “ancillary” model, as the site itself hosts more than 100000 open source projects and provides supporting services like mailing lists, source code versioning systems and file distribution. Also, companies that have a significant OSS contribution, but for which FLOSS is not the core business model were not included¹⁷.

¹⁵Among them: FreshMeat, Slashdot.org, OSNews, LinuxToday, NewsForge and some blog sites devoted to FLOSS business models like those of Roberto Galoppini, Matt Asay, Fabrizio Capobianco. Additional information was retrieved from Google searches.

¹⁶The original code for the SourceForge collaborative development environment was open source, and from its change of license several “forks” appeared, including Gforge.

¹⁷This for example includes IBM, HP and Sun; all of which are important FLOSS contributors, but for which open source software is just one of the overall revenue streams (along hardware, IT services and more).

	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 64 of 82
		Version: 1.0 Date: Oct. 10 2007
		Status : Final Confid : Public


An initial set of variables were selected, including: choice of licenses, product offering (whether a single version or multiple version of a software system are offered), services offered (divided into installation support, integration, training, consultancy, legal and technical certifications), type of contracts offered (subscriptions, licensing or per-incident) and metering form. Additionally, literature from each company's website was retrieved to find references to the business model adopted and how the model impacts the value proposition of the firm. Mailing lists and search engine searches were performed to obtain indicative references of the relationship of the company with the development community, and if there is an external, non-company based support activity in the form of websites, wikis and knowledge bases.

The collected data was then tabulated, eliminating non-significant variables; for example, coupling together installation, training, support and consulting that were found to be part of the offering of most of the companies that offered support services (and coupled in a single Installation/Training/Support/consulting variable, ITSC). The significant variables left are main revenue generation (the service or contractual offer that provides the main revenue to the company) and licensing model. The first is further subdivided into Selection services (finding appropriate FLOSS packages for a need), ITSC, subscription (a recurring license) and one-time licensing. The licensing model is obtained by looking at the licensing scheme adopted by the company and whether the company services were covering a single software project or a set of projects. By performing a simple cluster analysis on the results, it was possible to identify 6 main models and a "remainder" group:



Guide for SMEs
Deliverable ID: D8.1.1


	Company	Main Licensing model				Main revenue generation				
		dual licensing	OSS and commercial versions	Badgeware	Pure OSS	multiple packages covered	selection	ITSC	Subscription	licensing
dual lic.	Funambol	•						•		•
	Lustre	•						•		
	MuleSource	•		•					•	•
	Mysql	•							•	•
	OpenClovis	•							•	
	Pentaho	•						•		•
	sleepycatdb	•								•
Split OSS/commercial releases	Adaptive Planning		•							•
	Alterpoint		•					•		•
	Altinity		•					•		•
	Codeweaver (WINE)		•							•
	Coupa		•							•
	Digium (Asterisk)		•						•	
	Enormalism		•							•
	EnterpriseDB		•							•
	GreenPlum		•							•
	GroundWork		•						•	
	Hyperic		•						•	
	JasperSoft		•							•
	KnowledgeTree		•	•						
	OpenCountry		•							•
	Open-Xchange		•							
	NoMachine NX		•							•
	rPath		•						•	
	Scalix		•							•
	Sendmail		•							•
	Smoothwall		•							•
	Sourcefire (SNORT)		•							•
	Splunk		•							•
	SSLExplorer		•							•
SugarCRM		•	•						•	
TenderSystem		•	•						•	
VirtualBox		•							•	
Vyatta		•					•	•		
XenSource (Xen)		•					•			
Zend (PHP)		•							•	
ZIMBRA		•		•					•	
Badgeware	1bizcom			•				•		
	CATS applicant tracking			•					•	
	EmuSoftware/Netdirector			•				•	•	
	Jbilling			•				•		
	OpenBravo			•				•		
	OpenEMM			•				•		
	OpenTerracotta			•					•	
	SocialText			•						•
product specialists	Alfresco				•			•	•	
	Babel				•			•		
	CentraView				•			•		
	CleverSafe				•			•		
	Compiere				•			•		
	Exadel				•			•		
	Jitterbit				•			•		
	Mergere				•			•		
	Mindquarry				•			•		
	Mirth				•			•		
	OfBIZ				•			•		
	Qlusters (OpenQRM)				•			•		
	Symbiot/OpenSIMS				•			•		
	Talend				•			•		
	UltimateEMR				•			•	•	
	VISTA				•			•		
	vTiger				•			•		
Zenoss				•			•	•		
platf. Provid.	Jboss				•	•		•	•	
	RedHat linux				•	•		•	•	
	SourceLabs				•	•		•	•	
	SpikeSource				•	•	•	•	•	
	SUSE Linux				•	•		•	•	
	WSO2				•	•		•	•	
selection - consulting	ayamon					•	•	•		
	Enomaly					•	•	•		
	navica					•	•	•		
	openlogic					•	•	•		
	Optaros			•		•	•	•		
	x-tend					•	•	•		
Other	CiviCRM				•					
	Eclipse				•					
	Mozilla				•					
	OSAF Chandler				•					
	Sourceforge				•					

	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 66 of 82
		Version: 1.0
		Date: Oct. 10 2007
		Status : Final Confid : Public

The 6 main clusters identified are:

- **Dual licensing:** the same software code distributed under the GPL¹⁸ and a commercial license. This model is mainly used by producers of developer-oriented tools and software, and works thanks to the strong coupling clause of the GPL, that requires derivative works or software directly linked to be covered under the same license. Companies not willing to release their own software under the GPL can buy a commercial license that is in a sense an exception to the binding clause; by those that value the “free as in speech” idea of free/libre software this is seen as a good compromise between helping those that abide to the GPL and receive the software for free (and make their software available as FLOSS) and benefiting through the commercial license for those that want to maintain the code proprietary. The downside of dual licensing is that external contributors must accept the same licensing regime, and this has been shown to reduce the volume of external contributions (that becomes mainly limited to bug fixes and small additions).
- **Split OSS/commercial products:** this model distinguish between a basic FLOSS software and a commercial version, based on the libre one but with the addition of proprietary plugins. Most companies adopt as license the Mozilla Public License, as it allows explicitly this form of intermixing, and allows for much greater participation from external contributions, as no acceptance of double licensing is required. The model has the intrinsic downside that the FLOSS product must be valuable to be attractive for the users, but must also be not complete enough to prevent competition with the commercial one. This balance is difficult to achieve and maintain over time; also, if the software is of large interest, developers may try to complete the missing functionality in a purely open source way, thus reducing the attractiveness of the commercial version.


¹⁸An exception is MuleSource, that uses a MPL+Attribution license similar to the “badgeware” license described later. As the MuleSource CEO mentions, “So, if you use Mule in your software product and sell it commercially, then you are required to either make a licensing deal with us or keep the “powered by Mule” logo visible.” It is still debated by the community and experts if “badgeware” licenses are really open source; some of those have been submitted to the Open Source Initiative for evaluation.

	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 67 of 82
		Version: 1.0
		Date: Oct. 10 2007
		Status : Final Confid : Public

- **Badgeware:** a recent reinvention/extension of a previous license constraint¹⁹, that is usually based on the Mozilla Public License with the addition of a “visibility constraint”, the non-removability of visible trademarks or elements from a user interface. This allows the company to leverage trademark protection, and allows the original developers to receive recognition even if the software is resold through independent resellers.
- **Product specialists:** companies that created, or maintain a specific software project, and use a pure FLOSS license to distribute it. The main revenues are provided from services like training and consulting (the “ITSC” class) and follow the original “best code here” and “best knowledge here” of the original EUWG classification [DB 00]. It leverages the assumption, commonly held, that the most knowledgeable experts on a software are those that have developed it, and this way can provide services with a limited marketing effort, by leveraging the free redistribution of the code. The downside of the model is that there is a limited barrier of entry for potential competitors, as the only investment that is needed is in the acquisition of specific skills and expertise on the software itself.
- **Platform providers:** companies that provide selection, support, integration and services on a set of projects, collectively forming a tested and verified platform. In this sense, even linux distributions were classified as platforms; the interesting observation is that those distributions are licensed for a significant part under pure FLOSS licenses to maximize external contributions, and leverage copyright protection to prevent outright copying but not “cloning” (the removal of copyrighted material like logos and trademark to create a new product)²⁰. The main value proposition comes in the form of guaranteed quality, stability and reliability, and the certainty of support for business critical applications.


¹⁹The original BSD license introduced the “advertising claim”, that required the licensee to maintain in the advertising material mentioning feature or use of the software the wording “This product includes software developed by the University of California, Berkeley and its contributors”.

²⁰Examples of RedHat clones are CentOS and Oracle Linux.

	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 68 of 82
		Version: 1.0 Date: Oct. 10 2007
		Status : Final Confid : Public


- **Selection/consulting companies:** companies in this class are not strictly developers, but provide consulting and selection/evaluation services on a wide range of project, in a way that is close to the analyst role. These companies tend to have very limited impact on the FLOSS communities, as the evaluation results and the evaluation process are usually a proprietary asset.

The remaining companies are in too limited number to allow for any extrapolation, but do show that non-trivial business model may be found on ancillary markets. For example, the Mozilla foundation obtains a non trivial amount of money from a search engine partnership with Google (an estimated 72M\$ in 2006), while SourceForge/OSTG receives the majority of revenues from ecommerce sales of the affiliate ThinkGeek site; it is possible to classify those as “public funding” and “indirect funding” following the EUWG classification [DB 00].

	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 69 of 82
		Version: 1.0
		Date: Oct. 10 2007
		Status : Final Confid : Public


Bibliography

- [Aug 04] Augustin, L. *Living with open source: the new rules for IT vendors and consumers*. OSBC 2004 conference
- [Cam 06] Campbell, J. *Due Diligence at Eclipse: How it Benefits our Community*. EclipseCon 2006 presentation
- [Car 07] Carbone, P. *Value Derived from Open Source is a Function of Maturity Levels*, OCRI conference "Alchemy of open source businesses", 2007
- [CIO 07] CIOInsight, *CIOINSIGHT OSS survey 2007*.
- [COS 05] EU COSPA project, *D6.1 Report evaluating the costs/benefits of a transition towards ODS/OS*.
- [Cox 07] Cox, M. *Information sources*. Blog entry, <http://www.awe.com/mark/blog/200704101400.html>
- [DB 00] Daffara, C. Barahona, J.B. *Free Software/Open Source:*

	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 70 of 82
		Version: 1.0
		Date: Oct. 10 2007
		Status : Final Confid : Public


Information Society Opportunities for Europe? working paper,
<http://eu.conecta.it>

- [Daf 06] Daffara, C. *Sustainability of FLOSS-based business models*, II Open Source World Conference, Malaga 2006
- [Daf 06-2] Daffara, C. *Introducing open source in industrial environments*. 3rd CALIBRE workshop
- [Daf 07] Daffara, C. *Business models in OSS-based companies*. Accepted paper, OSSEMP workshop, Third international conference on open source. Limerick 2007
- [Dal 05] Dalle, J.-M., et al., *Advancing Economic Research on the Free and Open Source Software Mode of Production*, in *Building Our Digital Future: Future Economic, Social & Cultural Scenarios Based On Open Standards*, M. Wynants and J. Cornelis, Editors. 2005, Vrije Universiteit Brussels (VUB) Press: Brussels
- [ED 05] Evans Data, *Open Source Vision report*, 2005
- [EKM 05] Eckert D., Koch S., Mitlohner J. *Using the Iterated Prisoner's Dilemma for Explaining the Evolution of Cooperation in Open Source Communities* . Proceedings of the First International Conference on Open Source Systems , Genova 2005
- [Fed 07] Fedora Project Wiki, *Licensing*.
<http://fedoraproject.org/wiki/Licensing>
- [Fog 05] Fogel, K., *Producing Open Source Software: How to Run a Successful Free Software Project*. 2005: O'Reilly
- [Forr 07] Forrester consulting, *Open Source Software's Expanding Role in the Enterprise* . March 2007
- [Gar 06] Gartner Group, *Open source going mainstream*. Gartner report, 2006
- [Gosh 05] Gosh, et al. *Free/Libre/Open Source Software Worldwide impact study: FLOSSWorld*. FLOSSWorld project presentation.
<http://www.flossproject.org/papers/20051217/flossworld-intro3.pdf>
- [Gosh 06] Gosh, et al. *Economic impact of FLOSS on innovation and competitiveness of the EU ICT sector*.


	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 71 of 82
		Version: 1.0
		Date: Oct. 10 2007
		Status : Final Confid : Public

ec.europa.eu/enterprise/ict/policy/doc/2006-11-20-flossimpact.pdf


- [Hahn 02] Hahn, W.R. (editor), *Government policy towards open source software*. AEI-Brookings, 2002.
- [HSE 02] UK Health and Safety Executive, *Preliminary assessment of Linux for safety related systems*. Research report 011/2002
- [IBM 06] IBM, *Linux Client Migration Cookbook, Version 2: A Practical Planning and Implementation Guide for Migrating to Desktop Linux*. Available online at <http://www.redbooks.ibm.com/abstracts/sg246380.html?Open>
- [IDC 06] IDC, *Open Source in Global Software: Market Impact, Disruption, and Business Models*. IDC report, 2006
- [INES 06] INES IST project, *Final project report*.
http://www.euroines.com/down/INES_final_report.pdf
- [Inf 07] Infoworld white paper, *OPEN SOURCE MANAGEMENT: Trends, Requirements and Future Needs for the Open Source Enterprise*
- [Jaak 06] Jaaksi, A. *Building consumer products with open source*. LinuxDevices dec. 2006,
<http://www.linuxdevices.com/articles/AT7621761066.html>
- [Jul 06] Jullien N. (ed) *New economic models, new software industry economy*. RNTL report
- [KBST 06] Germany KBSt, *Migration guide*.
- [Kli 05] Klinecicz, K. *Innovativeness of open source software projects*. Technical report, School of Innovation Management, Tokyo Institute of Technology. 2005
- [Mue 07] Mueller, M. *Openoffice.org projects by Members*,
http://blogs.sun.com/GullFOSS/entry/openoffice_org_projects_by_members
- [OECD 02] OECD, "OECD/Eurostat task force on software measurement in the national accounts", Conference of European Statisticians, Joint ECE/Eurostat/OECD meeting on national accounts, 2002

	Guide for SMEs Deliverable ID: D8.1.1	Page : 72 of 82
		Version: 1.0
		Date: Oct. 10 2007
		Status : Final Confid : Public

- [Ost 04] Osterwalder A, *The business model ontology – a proposition in a design science approach* , PhD Dissertation, University of Lausanne, Switzerland, 2004
- [Ost 05] Osterwalder A, Pigneur Y., *E-business models and disruptive behaviours*,
<http://www.businessmodeldesign.com/presentations/PACIS05.ppt>
- [QSOS 06] QSOS project, *Method for Qualification and Selection of Open Source software (QSOS) version 1.6.*
- [Raym 00] Raymond, E.S., *The Cathedral and the Bazaar*, in *The Cathedral and the Bazaar*. 2000
- [Reas 06a] Reasoning Inc. *A Quantitative Analysis of TCP/IP Implementations in Commercial Software and in the Linux Kernel.*
- [Reas 06b] Reasoning Inc. *How Open Source and Commercial Software Compare: Database Implementations in Commercial Software and in MySQL.*
- [Rig 06] Rigby P.C., German D.M. *A preliminary examination of code review processes in open source projects.* University of Victoria technical report, 2006,
<http://opensource.mit.edu/papers/Rigby2006TR.pdf>
- [Ros 05] Rosen L., *Open source licensing*. Prentice Hall, 2005
- [Sei 06] Seigo A., *The quest for project identity and definition. Keynote speech*, Akademy conference 2006.
- [Sch 02] Schiff, A. *The Economics of Open Source Software: A Survey of the Early Literature.* Review of Network Economics, 1 (1), March 2002
- [Spi 02] Spiller, D. and T. Wichmann, *FLOSS 3: Basics of Open Source Software Markets and Business Models*, in *FLOSS – Free/Libre Open Source Software: Survey and Study*. Berlecon Research, Berlin 2002
- [Stu 07] Stuermer, M. *How money influences open source projects and its contributors.* LinuxTag 2007, Berlin.

	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 73 of 82
		Version: 1.0 Date: Oct. 10 2007
		Status : Final Confid : Public

- [Suc 04] Succi, Paulson, Eberlein. *An Empirical Study of Open-Source and Closed-Source Software Products*, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, V.30/4, april 2004
- [Sun 06] Sun Microsystems, *Free and Open Source licensing*. White paper, www.sun.com/software/opensource/whitepapers/Sun_Microsystems_OpenSource_Licensing.pdf
- [UUS 05] Ueda, M., Uzuki T., Suematsu C. *A cluster analysis of open source licenses*. Proceedings of the First International Conference on Open Source Systems, Genova 2005
- [VH 03] Von Hippel, E. and G. von Krogh, *Open Source Software and the "Private-Collective" Innovation Model: Issues for Organizational Science*. Organization Science, 2003. 14(2): p. 209-223
- [VH 05] Von Hippel, E. *Democratizing innovation*. MIT press, 2005

	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 74 of 82
		Version: 1.0 Date: Oct. 10 2007
		Status : Final Confid : Public

Appendix 1: estimating the number of active FLOSS projects

A recurring debate discussion among FLOSS-supporters and detractors is related to the estimation of the real number of active FLOSS projects. While it is easy to look at the main repository site (sourceforge.net) that boasts more than 100.000 projects, it is equally easy to look in more depth and realize that a significant number of those projects are really abandoned or have no significant development.

For the purpose of obtaining some unbiased estimates, we performed a first search among the main repository sites and FLOSS announce portals; we also set a strict activity requirement, stately an activity index from 80 to 100% and at least a file release in the last 6 months. Of the overall 155959 projects, only 10656 (6.8%) are "active" (with a somehow very restrictive definition; a more relaxed release period of 1 year shows an active percentage of 9.2% or 14455 projects).

However, while Sourceforge can rightly be considered the largest single repository, it is not the only potential source of projects; there are many other vertical repositories, among them BerliOS, Savannah, Gna! and many others, derived both from the original version of the Sourceforge code and many more based on a rewritten version called GForge.²¹

The result summary is:

Repository name	Number of projects
All GForge sites ²²	16776
Berlios Sourcewell	3340
Savannah	2793
Gna!	1039

That gives a total of 23948 projects, to which (using a sampling of 100 projects from each) we have found a similar number of active projects (between 8% and 10%).


The next step is the estimation of how many projects of the overall FLOSS landscape are hosted on those sites, and for performing this estimate we took the entire FreshMeat²³ announce database, as processed by the FLOSSmole project²⁴ and found that the projects that have an homepage in one of the repository sites are 23% of the total. This count is however biased by the fact that the probability of a project to be announced on FreshMeat is not equal for all projects; that is, english-based and oriented towards a large audience have a much higher probability to be listed. To take this into account, we performed a search for non-english based forges, and for software that is oriented towards a very specific area, using data from past IST projects like Spirit and AMOS. We have found that non-english

²¹It has been suggested to the authors that in this way we can end up counting twice those projects that move from one site to others. The reality is that as the "old" project becomes inactive, it is removed from the count and so this risk is limited to those that performed the move in the last 12 months only (as moving is rather uncommon, this is however a very small number that should not influence the overall percentages).

²²As reported in the GForge site count, <http://gforge.org/docman/view.php/1/52/gforge-sites.html>

²³A popular FLOSS announcement portal. www.freshmeat.net


²⁴a collaborative collection and analysis of FLOSS data, <http://ossmole.sourceforge.net/>

	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 76 of 82
		Version: 1.0
		Date: Oct. 10 2007
		Status : Final Confid : Public

projects are underrepresented in FreshMeat in a significant way, but as the overall "business-readiness" of those projects is unclear (as for example there may be no translations available, or be specific to a single country legal environment) we have ignored them. Vertical projects are also underrepresented, especially with regard to projects in scientific and technical areas, where the probability of being included is around 10 times lower compared to other kind of software. By using the results from Spirit, a sampling from project announcements in scientific mailing lists, and some repositories for the largest or more visible projects (like the CRAN archive, that hosts libraries and packages for the R language for statistics, that hosts 1195 projects) we have reached a lower bound estimate of around 12000 "vertical" and industry-specific projects.

So, we have an overall lower bound estimate of around 195000 projects, of which we can estimate that 7% are active, leading to around 13000 active projects. Of those, we can estimate (using data from Slashdot, FreshMeat and the largest Gforge sites) that 36% fall in the "stable" or "mature" stage, leading to a total of around 5000 projects that can be considered suitable for an SME, that is with an active community, stable and with recent releases.

It should be considered that this number is a lower bound, obtained with slightly severe assumptions; also, this estimate does not try to assess the number of projects not listed in the announcement sites (even vertical application portals); this is a deliberate action, as it would be difficult to estimate the reliability of such a measure, and because the "findability" of a project and its probability of having a sustained community participation are lower if it is difficult to find information on the project in the first place; this means that the probability of such "out of the bounds" projects would probably be not a good opportunity for SME adoption in any case.


	<p style="text-align: center;">Guide for SMEs</p> <p style="text-align: center;">Deliverable ID: D8.1.1</p>	Page : 77 of 82
		Version: 1.0
		Date: Oct. 10 2007
		Status : Final Confid : Public

Appendix 2: QSOS assessment score tables


The data provided is a synthesis of the official QSOS assessment methodology in its 1.6 revision, available from www.qsos.org, along with several useful tools to facilitate the measurement and score collection steps. The axis of evaluation includes criteria to estimate risks incurred by the user when adopting free or open source software. Scoring of criteria is done independently of any particular user's context (the context is considered later in Step 3 - "Qualification"); criteria are split into five categories:

- Intrinsic durability
- Industrialized solution
- Integration
- Technical adaptability
- Strategy


After a "generic" part, depending on the application area it is possible to create custom QSOS sheets; in the following example a "groupware" evaluation is provided.

	Guide for SMEs Deliverable ID: D8.1.1	Page : 78 of 82
		Version: 1.0
		Date: Oct. 10 2007
		Status : Final Confid : Public

Criterion	Score 0	Score 1	Score 2
Generic section			
Intrinsic durability			
Maturity			
Age	less than 3 months	if between 3 months and 3 years	after 3 years
Stability	Unstable software with numerous releases or patches generating side effects	Stabilized production release existing but old. Difficulties to stabilize developpement releases	Stabilized software. Releases provide bug fixes corrections but mainly new functionalities
History	Software knows several problems which can be prohibitive	No know major problem or crisis	History of good management of crisis situations
Fork	Software is very likely to be forked in the future	Software comes from a fork but has very few chances of being forked in the future	Software has very little chance of being forked. It does not come from a fork either
Adoption			
Popularity	Very few users identified	Detectable use on Internet	Numerous users, numerous references
References	None	Few refences, non critical usages	Often implemented for critical applications
Contributing Community	No community or without real activity (forum, mailing list, ...)	Existing community with a notable activity	Strong community: big activity on forums, numerous contributors and advocates
books	No book about the software	Less than 5 books about the software are available	More than 5 books about software are available, in several languages
Development leadership			
Leading team	1 to 2 individuals involved, not clearly identified	Between 2 and 5 independent people	More than 5 people
Management style	Complete dictatorship	Enlightened despotism	Council of architects with identified leader (e.g: KDE)
Activity			
Developers, identification, turnover	Less than 3 developers, not clearly identified	Between 4 and 7 developers, or more unidentified developers with important turnover	More than 7 developers, very stable team
Activity on bugs	Slow reactivity in forum or on mailing list, or nothing regarding bug fixes in releases note	Detectable activity but without process clearly exposed, loing reaction/resolution time	Strong reactivity based on roles and tasks assignment
Activity on functionalities	No or few new functionalities	Evolution of the product driven by the core team or by user's request without any clearly explained process	Tool(s) to manage feature requests, strong interaction with roadmap
Activity on releases	Very weak activity on both production and development releases	Activity on production and developmenet releases. Frequent minor releases (bug fixes)	Important activity with frequent minor releases (bugs fixes) and planned major releases relating to the roadmap forecast


	Guide for SMEs Deliverable ID: D8.1.1	Page : 79 of 82
		Version: 1.0 Date: Oct. 10 2007
		Status : Final Confid : Public

Criterion	Score 0	Score 1	Score 2
Generic section			
Industrialized solution			
Independence of developments	Developments realized at 100% by employees of a single company	60% maximum	20% maximum
Services			
Training	No offer of training identified	Offer exists but is restricted geographically and to one language or is provided by a single contractor	Rich offers provided by several contractors, in several languages and split into modules of gradual levels
Support	No offer of support except via public forums and mailing lists	Offer exists but is provided by a single contractor without strong commitment quality of services	Multiple service providers with strong commitment (e.g: guaranteed resolution time)
Consulting	No offer of consulting service	Offer exists but is restricted geographically and to one language or is provided by a single contractor	Consulting services provided by different contractors in several languages
Documentation	No user documentation	Documentation exists but shifted in time, is restricted to one language or is poorly detailed	Documentation always up to date, translated and possibly adapted to different target readers (end user, sysadmin, manager, ...)
Quality Assurance			
Quality Assurance	No QA process	Identifies QA process but not much formalized and with no tool	Automatic testing process included in code's life-cycle with publication of results
Tools	No bug or feature request management tool	Standard tools provided (for instance by a hosting forge) but poorly used	Very active use of tools for roles/tasks allocation and progress monitoring

	Guide for SMEs Deliverable ID: D8.1.1	Page : 80 of 82
		Version: 1.0
		Date: Oct. 10 2007
		Status : Final Confid : Public

Criterion	Score 0	Score 1	Score 2
Generic section			
Packaging			
BSD			
FreeBSD	The software is not packaged for FreeBSD	A port exists but it has important issues or it doesn't have official support	A official port exists in FreeBSD
Mac OS X	The software is not packaged for Mac OS X	A package exists but it has important issues or it doesn't have official support	The software is packaged in the distribution
NetBSD	The software is not packaged for NetBSD	A port exists but it has important issues or it doesn't have official support	A official port exists in NetBSD
OpenBSD	The software is not packaged for OpenBSD	A port exists but it has important issues or it doesn't have official support	A official port exists in OpenBSD
Linux			
Debian	The software is not packaged for Debian	A Debian package exists but it has important issues or it doesn't have official support	The software is packaged in the distribution
Mandriva	The software is not packaged for Mandriva	A package exists but it has important issues or it doesn't have official support	The software is packaged in the distribution
Red Hat	The software is not packaged for Red Hat/Fedora	A package exists but it has important issues or it doesn't have official support	The software is packaged in the distribution
SuSE	The software is not packaged for SuSE	A package exists but it has important issues or it doesn't have official support	The software is packaged in the distribution
Source	Software can't be installed from source without lot of work	Installation from source is limited and depends on very strict conditions (OS, arch, lib, ...)	Installation from source is easy
Unix			
AIX	The software is not packaged for AIX	A package exists but it has important issues or it doesn't have official support	A stable package is provided for AIX
HP-UX	The software is not packaged for HP-UX	A package exists but it has important issues or it doesn't have official support	A stable package is provided for HP-UX
Solaris	The software is not packaged for Solaris	A package exists but it has important issues or it doesn't have official support (e.g: SunFreeware.com)	The software is supported by Sun for Solaris
Windows	The project can't be installed on Windows	A package exists but it is limited or has important issues or just cover some specific Windows release (e.g: Windows 2000 and Windows XP)	Windows is full supported and a package is provided

Criterion	Score 0	Score 1	Score 2
Generic section			
Exploitability			
Ease of use, ergonomics	Difficult to use, requires an in depth knowledge of the software functionality	Austere and very technical ergonomics	GUI including help functions and elaborated ergonomics
Administration / Monitoring	No administrative or monitoring functionalities	Existing, functionalities but uncomplete and or need improvement	Complete and easy-to-use administration and monitoring functionalities. Possible integration with external tools (e.g: SNMP, syslog, ...)
Technical adaptability			
Modularity	Monolithic software	Presence of hight level modules allowing a first level of software adaptation	Modular conception, allowing easy adaptation of the software by selecting or creating modules
Code modification	Everything by hand	Recompilation possible but complex without any tools or documentation	Recompilation with tools (e.g: make, ANT, ...) and documentation provided
Code extension	Any modification requires code recompilation	Architecture designed for static extension but requires recompilation	Principle of plugin, architecture designed for dynamic extension without recompilation
Strategy			
License			
Permissiveness	Very strict license, like GPL	Moderate permissive license located between both extremes (GPL and BSD) dual-licensing depending on the type of user (person, company, ...) or their activities	Very permissive like BSD or Apache licenses
Protection against proprietary forks	Very permissive like BSD or Apache licenses	Moderate permissive license located between both extremes (GPL and BSD), dual-licensing depending on the type of user (person, company, ...) or their activities	Very strict license, like GPL
Copyright owners	Rights held by a few individuals or entities, making it easier to change the license	Rights held by numerous individuals owning the code in a homogeneous way, making relicense very difficult	Rights held by a legal entity in whom the community trusts (e.g: FSF or ASF)
Modification of source code	No practical way to propose code modification	Tools provided to access and modify code (like CVS or SVN) but not really used to develop the software	The code modification process is well defined, exposed and respected, based on roles assignment
Roadmap	No published roadmap	Existing roadmap without planning	Versionned roadmap, with planning and measure of delays
Sponsor	Software has no sponsor, the core team is not paid	Software has an unique sponsor who might determine its strategy	Software is sponsored by industry
Strategical independence	No detectable strategy or strong dependency on one unique actor (person, company, sponsor)	Strategical vision shared with several other free and open source projects but without strong commitment from copyrights owners	Strong independence of the code team, legal entity holding rights, strong involvement in the standardization process

	Guide for SMEs Deliverable ID: D8.1.1	Page : 82 of 82
		Version: 1.0
		Date: Oct. 10 2007
		Status : Final Confid : Public

Criterion	Score 0	Score 1	Score 2
groupware-specific Administration GUI			
Web interface	No web interface	an web interface is provided but limited	everything can be completed with the Web interface
Console mode	Nothing	Some tools exists, but limited. No text based configuration file or not human readable (e.g: complexe XML)	Total access to the server configuration with powerful tools and well designer text configuration file
Stand alone admin tool	Nothing	A limited tool exist allow user to do specific operation	A powerful tool give access to every major features of the server
Supported groupware			
Calendar	no calendar provided	a calendar is provided, but leak some important features	a well integrated calendar is provided
Taskmanager	no task manager provided	a task manager is provided but leak some important features	Task manager fully supported
Notemanager	no note manager provided	a limited note manager is provided	a well integred note manager is provided
Contact manager	can't add contact in the server	contact manager exists but is limited	contact manager fully supported
Standard support			
iCalendar over WebDav	iCalendar over WebDav is not supported	iCalendar over WebDav is partially supported	iCalendar over WebDav works
CalDav	CalDav is not supported	CalDav is partially supported	CalDav works
Groupdav	Groupdav is not supported	Groupdav is partially supported	Groupdav works
SyncML	SyncML is not supported	SyncML is partially supported	SyncML works
Supported client			
Web client	Web interface doesn't exist	Web interface is provided but limited or need some work for its integration	Web interface directly provided with the project
Microsoft Outlook	Microsoft Outlook not supported	Microsoft Outlook connector is provided but have some limitation	a free Microsoft Outlook connector
Novell Evolution	Novell Evolution can't be used	Novell Evolution can be used but with some limitation	Novell Evolution fully supported
KDE	KDE PIM (Korganizer, kmail, ...) can't be used with this groupware	KDE can be used but with some limitation	Fully support of KDE
Apple iCal	Apple's iCal can't be used	Apple's iCal works but with some limitation	Apple's iCal fully supported
Performance			
Load balancing	This software can't be loadbalancer	Part of the installation can be splitted but it keeps important bottleneck	Loadbalancing just works
Code quality			
Remote access API	No remote remote API	remote API (SOAP, XML/RPC, REST) exists but is limited or buggy	powerful remote API (SOAP, XML/RPC, REST) provided
unified API	No API provided to extend the server, or very limited and not documented	An API is provided but limited of not fully documented	Well documented and complet API