ITEA Report on Open Source Software

January, 2004

ITEA

INFORMATION TECHNOLOGY

FOR EUROPEAN ADVANCEMENT

Foreword

As ITEA grows and unfolds, the numbers of its stakeholders are steadily growing too and their business interests are becoming increasingly diverse. In some subject areas, e.g. in the main technological directions which we encourage, the road-mapping work[1] is key, and regular updates of the ITEA Roadmap are published. Other major inputs are the IRIS Book (Interim Report on ITEA's Status)[2] and our annual symposium provides an opportunity to wrap up developments that have taken place over the previous year.

This document – the first special report from ITEA – is of a somewhat different nature. It analyses the current position of Open Source Software, examining this unique phenomenon from many angles, including technological, economic and legal aspects.

Why this report?
There's a great deal of confusion and hype around Open Source Software. There's even a lack of unanimity about what to call it. For example, you come across terms like "Open Software", "Open Source Software", "Free Software" and "Freeware", which are used interchangeably. It's also clear that the potential value of Open Source Software for industry and the scientific community depends on the specific underlying business model or application. This explains some of the differences. Technical and non-technical factors, as well as features related to the software life cycle (e.g. updates, performance, reliability and adaptability) vary from one situation to another, depending to a large extent on the "owner" of the technology. Another critical concern from the point of view of industry is the legal position (ranging from protection of Intellectual or Industrial Property Rights to liability issues).

These considerations more or less scoped the way this report was produced. A group of Open Source experts from ITEA companies (for the full list, see Appendix 6) worked together to produce a report that examines the technical, business and legal problems associated with Open Source Software from the perspective of those involved in developing and exploiting embedded and distributed software.

You can read the results of this investigation in the pages that follow. I trust that you will find this contribution of value and relevant to your perspectives.

Jean-Pierre Lacotte
Vice-chairman of ITEA

[1] The latest version of ITEA's Technology Roadmap for Software-Intensive Systems can be downloaded free of charge from our website: www.itea-office.org

[2] The IRIS Book can be downloaded free of charge from our website: www.itea-office.org

Table of contents

Foreword **3**

1 Introduction **7**

2 An overview of OSS from the point of view of industry **8**

2.1 A new development paradigm 8
2.1.1 Key features 8
2.1.2 Main issues 8
2.2 Specific business opportunities 9
2.2.1 Key features 9
2.2.2 Main issues 9
2.3 Thorny issues: patents and licences 9
2.3.1 Key features 10
2.3.2 Legal issues 10
2.4 Risks and threats vs. advantages and opportunities 10
2.5 Consequences 12
2.6 Conclusion 12

APPENDICES

A1 Technical aspects of OSS **15**

A1.1 New development paradigm 15
A1.1.1 Change in the content of contributions from academia 15
A1.1.2 It's progress, but it's not necessarily THE solution 15
A1.2 OSS in the various technology categories of the ITEA roadmap 16
A1.3 OSS and quality 16
A1.3.1 What does quality mean for industry? 16
A1.3.2 What has been achieved so far with OSS? 17
A1.3.3 An industry perspective on quality 18
A1.3.4 Areas for improvement 18
A1.4 Research directions 19

A2 Economic and industrial aspects of OSS **20**

A2.1 Software in the European Market: numbers and trends 20
A2.2 OSS and the software industry 23
A2.2.1 With OSS, free gets a value 23
A2.2.2 OSS allows shared R&D 23
A2.2.3 OSS enables reusability 24
A2.2.4 OSS reinforces sustainability 24
A2.2.5 OSS and business 24
A2.3 From utility to embedded software 25

A3 Legal implications of OSS 27

A3.1 Introduction 27

A3.2 Legal issues 27

A3.2.1 Copyright and licensing 27

A3.2.2 Liability 28

A3.2.3 Patents 28

A3.3 Risks when using Open Source Software 29

A3.3.1 The architectual level 29

A3.3.2 The business level (patent risks) 29

A3.4 Conclusion 30

A4 References 32

A5 Glossary 33

A6 Members of the ITEA Open Source Software Working Group 34

1 Introduction

Open Source Software (OSS) is neither a new technology nor a new method for developing software. Some decades ago (old *geeks* do remember!), lots of software used to be free. The history of Open Source is most closely linked to the creation of the first Unix by Bell Labs back in 1969.

As an injunction by the U.S. Department of Justice prevented AT&T selling software, they provided numerous licensees with the code and manuals, but no support. As the numbers of Unix users grew, an Open Source movement developed around the University of California in Berkeley and, in 1977, the first Berkeley Software Distribution (BSD) of Unix was established.

X Window System is another major OSS project. It originated at MIT, with the involvement of IBM, DEC and HP, among other well-known firms. In the same period – the early 1980s – the GNU project (a free development environment that is compatible with proprietary Unix commands) started, also at MIT. The GNU project is at the heart of two interesting formations: the Free Software Foundation (FSF) and the General Public License (GPL). During the 1990s the most famous examples of OSS were Linux and Apache.

Today, the OSS movement is getting wide recognition in the software arena, raising hopes and expectations on the one hand, while generating caution and mistrust on the other, and it is at risk of becoming another ground for quasi-religious conflicts between those campaigning for and against it. Numerous projects are being launched, some of which bring together very different, but important players[3], signalling considerable interest from industry.

An ITEA Core Team containing people from a wide range of backgrounds chose this moment in time to write a report on Open Source Software. There are a number of reasons for this:

- most companies have set internal rules to cope with the use of OSS components
- in some applications OSS appears to be a valuable substitute for traditional software
- OSS has gained such momentum that in many countries around the world – as well as in the European Commission – public authorities are increasing their support for the OSS movement
- some big companies now view OSS as a major competitor.

A number of extremely valuable reports, papers, books and articles on OSS have already been written. Much of the content of this report is based on these sources and they are, wherever possible, referenced. This report reflects an **industry-driven** view on OSS. As ITEA's sphere of activity is pre-competitive R&D in **embedded and networked software**, this report naturally focuses on this type of software. This approach is, in fact, rather different from that taken by most users and promoters of OSS.

This paper presents the current situation from three different angles (*Technical, Economy, Legal*) and for each of them it highlights a number of specific points. Readers should be aware that the distinction between these different points of view is made merely for the sake of clarification. None is in fact independent of the others.

Three appendices subject the arguments outlined briefly below to more detailed analysis.

Before we present the core arguments in this report we need to clarify what we mean by *Open Source Software*. This formulation has been chosen because it is the one that is most widely used. The term "freeware" has been deliberately avoided, partly because it relates to hardware as well as to software. Since ITEA is a software programme, we have restricted this report to software aspects.

Then, we have used the simplest definition we could find: OSS is a collective name for all kinds of software for which the source code is freely available. Anyone may extend, improve and distribute such software. This is another reason not to use the term "freeware", a concept that is directly linked to money, while Open Source is not. Most OSS licenses are different, since charging a fee is not even an issue. One can be compliant with part of an OSS license even when charging money for a licensed piece of software.

[3] For example, the Videolan (www.videolan.org) project groups IBM, SNCF and AT&T (among others) with the French *Ecole Centrale*.

2 An overview of OSS from the point of view of industry

We distinguish three main features of OSS:
- From a technical point of view, we see a *new development paradigm*
- From an economic point of view, we can identify some *specific business opportunities*
- From a legal point of view, we recognise some particularly *thorny issues* related to patents and Intellectual Property.

We have deliberately decided to present these three approaches separately and, in this part of the paper, to keep that presentation as brief as possible so as to highlight only what we consider to be the **most important points**. Detailed analyses of each of the subjects dealt with here can be found in Appendices 1, 2 and 3 of which this section is a summary. They contain nuances for which there is no space here, where we restrict ourselves to the most salient aspects of the OSS landscape.

Each of the subsections contains two parts. The first consists of a brief overview of the "state of the art" in an attempt to identify the main characteristics of OSS. The second raises some industry concerns. These characteristics and issues are the ones that we consider to be of **crucial importance** within the context of ITEA. We do not pretend to provide an exhaustive review of the rich and complex world of OSS.

2.1 A new development paradigm

The main technical features of OSS are closely related to the origins of the OSS movement: research labs in companies and universities exchanging programs and information between different locations. Using OSS in an industrial environment of embedded and networked software immediately raises questions that are related to the quality of the resulting system, product or service.

2.1.1 Key features

The development of a piece of OSS has two distinctive features:
- It is a collaborative, distributed process, while most system development today is isolated and centralized
- A number of developments and enhancements originate from academia or research organizations.

Let us examine these points in turn.

- **Collaborative and distributed vs. singular and centralized**
  From a technological point of view, the industrial quality of a piece of softwa re can be roughly assessed by its reliability[4]. Often, the quality of OSS has been shown to be excellent. But the methods used to achieve it are far from usual industry practice.
  → Quality depends on the **number** of "developers/users": there is no formal process, rather a generalized (and very efficient) development cycle based on trial and error. The more popular a piece of OSS is, the more it will be debugged and reliable.
  → In turn this **number** is influenced by the interest of people in the software being developed. In a dramatic (and not exact) image, one could say that, at the time, there is one category of industrial grade software that is supported by strong development communities: i.e. "commodity" software (e.g. Linux, GNU and Apache).

- **In OSS development, academic developers are more directly involved in creating new industrial standards.**
  In most examples of successful OSS ventures, there has been a heavy investment by academia and, more generally, by research organizations.
  → This is an **extraordinary opportunity** for Europe to make the most of the invaluable pool of (human) resources in technical educational institutions
  → However, there is a risk that some subjects, which would be of great interest to industry, are felt to be uninteresting by this part of the OSS community and are therefore neglected.

2.1.2 Main issues

This quick overview of the technical aspects of OSS brings us to address the following issues, which could be of great importance:

[4] Please refer to Appendix 1 for a more detailed discussion of this aspect

- **Several areas that are important for embedded and distributed software are still not covered**
  As explained in Appendix 1, there are some gaps with regard to OSS in the technology categories highlighted in the ITEA Roadmap[5]. Unlike much commodity software (e.g. operating systems), the platforms and middleware required by most ITEA stakeholders (e.g. real-time applications or software designed for industries like aeronautics) do not have wide support, especially in the universities. This restricts the number of opportunities for building solutions based on OSS.

- **Quality assurance**
  Many applications, whether products, systems or services, that are based on software developed within the ITEA realm are time-critical or safety-critical. It is very difficult to use OS middleware or platforms to build these applications because of the lack of quality assurance and/or certification. As emphasised in Appendix 2, this situation hampers the use of OSS versus Commercial-Off-The-Shelf (COTS). Appendices 1 and 2 present some possible solutions to this problem.

2.2 Specific business opportunities

Starting from (almost) non-commercial premises, OSS has proved to be a valuable tool for business. In some cases, it appears to be a life-cycle cost-breaker for manufacturer and customer alike[6]. In other cases, it can be used to re-establish competition in an area that was previously dominated by a monopoly supplier[7]. This might suggest alternative uses for certain middleware, which is the key to the convergence of IT, consumer electronics and telecom in industry, when looked at from an ITEA perspective. For a more detailed analysis of this issue, see Appendix 1 & 2.

2.2.1 Key features

- **The cost aspect**
  ` There are two arguments that explain the cost advantage of OSS.
  → Reusability: as the same components are used a number of times by many different actors, this cuts the cost of **development**, which, in turn, means that the costs of verification are shared, thus reducing set-up time.
  → Reduced license fees: OSS components are **widely used at minimal cost**. Although OSS is not entirely free, the low cost of licenses brings down the price of the most commonly used components, thus reducing the price of the applications that they power.

- **The standards aspect**
  Cost reduction is not the only business strategy. Creating a widely-accepted standard is another. Here the co-operative development feature of OSS plays a key role.
  → Creating, using and promoting OSS in software products can help those fighting uphill battles against companies with dominant positions in the market or to create new business opportunities (for prime examples of this, just reflect on the development of Linux, Apache and MySQL).

2.2.2 Main issues

The convergence battle is currently raging, and European industry is not always very well placed to win[8]. To remain in control of the ever-increasing role that networking will play in the daily life of Europe's citizens, there are some key middleware products and platforms that simply cannot be left to monopolies, wherever they are based. Among these, the most salient right now are those related to safety, security and privacy.

There is a clear need for Open Source alternatives in some middleware areas that are crucial to convergence, such as trusted computing[9]. Such alternatives should involve industries and universities worldwide.

2.3 Thorny issues: patents and licenses

It is not only in the area of development that OSS presents new opportunities for industry. The numerous issues related to Intellectual (or Industrial) Properties and responsibilities, although at core the same are in practice very different, mainly because of the current licensing framework. In some cases, when redistributing a piece of OSS, the user may have to transmit as OSS everything originally

[5] However, some areas have been covered, for example, during our 2002 Symposium the ITEA Achievement Award was won by the PEPiTA project, which includes development of part of the OSS platform Jonas.

[6] A recent example of this is the choice of Open Source solutions by the city of Munich for its IT infrastructure.

[7] E.g. the success of Apache, which created a unique business opportunity for some players in the server arena (especially IBM).

[8] For more on this subject, see ITEA's IRIS Book.

[9] Trusted Computing provides a computing platform on which you can't tamper with the application software, and where these applications can communicate securely with their authors and with each other [Anderson 2003].

received as OSS, plus all enhancements made subsequently. This subject is explained in detail in Appendix 3. In essence, the industrial user of OSS has to deal with two rather difficult issues:

- conflicts between IP and OSS, i.e. the problem of losing the right to assert patents, when a product is implemented in OSS
- possible patent infringement.

As difficult as these issues may seem, they have been nevertheless been solved satisfactorily in most cases as the growing number of corporations engaged in the use and/or development of OSS testifies. Let us briefly examine the main points.

2.3.1 Key features

We focus here on two key aspects of the industrial use of OSS: can you protect your Intellectual Property Rights (IPR)? And then there's the issue of liability?

- **OSS vs. IPR**
  This conflict powerfully brings together the technical and legal sides of the use of OSS. This is because, with some of the earliest licenses, it was very difficult to protect company know-how and IP.
  → Today, there are many different kinds of licenses to choose from, each offering a different level of protection. One important factor to bear in mind is the fact that the type of license chosen will strongly depend on the specific business goal and on the value of the IPR of the company.
  → Architectural design is of prime importance. Since all of these licenses (except for BSD) make a clear difference between software that is "based on" OSS and the rest, the route to securing one's IPR is through careful architectural design of the application that uses the OSS.

- **Liability:**
  There are two main sides to liability: the legal one is related to patents and copyright; the other, which we call industrial responsibility, is related to the consequences of improper use of a system, product or service that incorporates a piece of OSS.
  → **Legal**: software patents vs. the OS community. In most cases the OSS community does not pay much attention to software patents. As a result the developers of OSS do not care too much whether or not they infringe an existing software patent when they develop a piece of software. Another legal aspect is the possibility that a product based on open source software may infringe on the copyright of a third party.
  → **Industrial responsibility**: COTS vs. OSS
  As the correct behaviour of a piece of software can be linked to the supplier of specific COTS, industrial responsibility can be easily traced in cases of failure. However, when a piece of OSS is used, the total responsibility lies with whoever assembles the system.

2.3.2 Legal issues

Legal issues may appear to be the most important factors hampering the rapid advancement of the use of OSS in industry.

- It's important to clearly discriminate between the different types of risks and to clarify precisely where the legal border lies between fair and unfair use of OSS. This is especially true where technologies are evolving rapidly and, as a result, dynamic connections between two pieces of software will become the norm. This will increasingly blur the meaning of the term "based on", the present touchstone for identifying possible infringements of the OSS rules.

2.4 Risks and threats vs. advantages and opportunities

This chapter highlights the important role that OSS is presently playing in commodity software, from the point of view of industry. Special attention has been paid to the fact that systems integrators (in other words, most of the companies involved in manufacturing and selling products or systems containing embedded software) may also reap important benefits from OSS, provided they adapt their organisation to its use. For any organisation being involved in the OSS community means broadening the scope of expertise that they are able to tap into for software technologies and elements that do not infringe on their value added know-how and knowledge. This process is broadly described in Figure 1.

When a company is not part of the OSS movement, the process for developing new systems, products and services basically creates a loop between R&D and proprietary applications. On the drawing, this is represented by arrows 2 and 3, which connect R&D to proprietary applications. These applications are linked to a repository of systems, products and services (on the right-hand side of the picture), which, in turn feeds back challenges to R&D.

development time, and to avoid dependence on a single vendor. It is therefore to be expected that more and more companies and institutions will start using open source software. There are, however, some risks associated with doing so. Being forced to release some or all of the software of a commercial product as open source software may greatly reduce its value. And the risk to one's patent portfolio should not be underestimated.

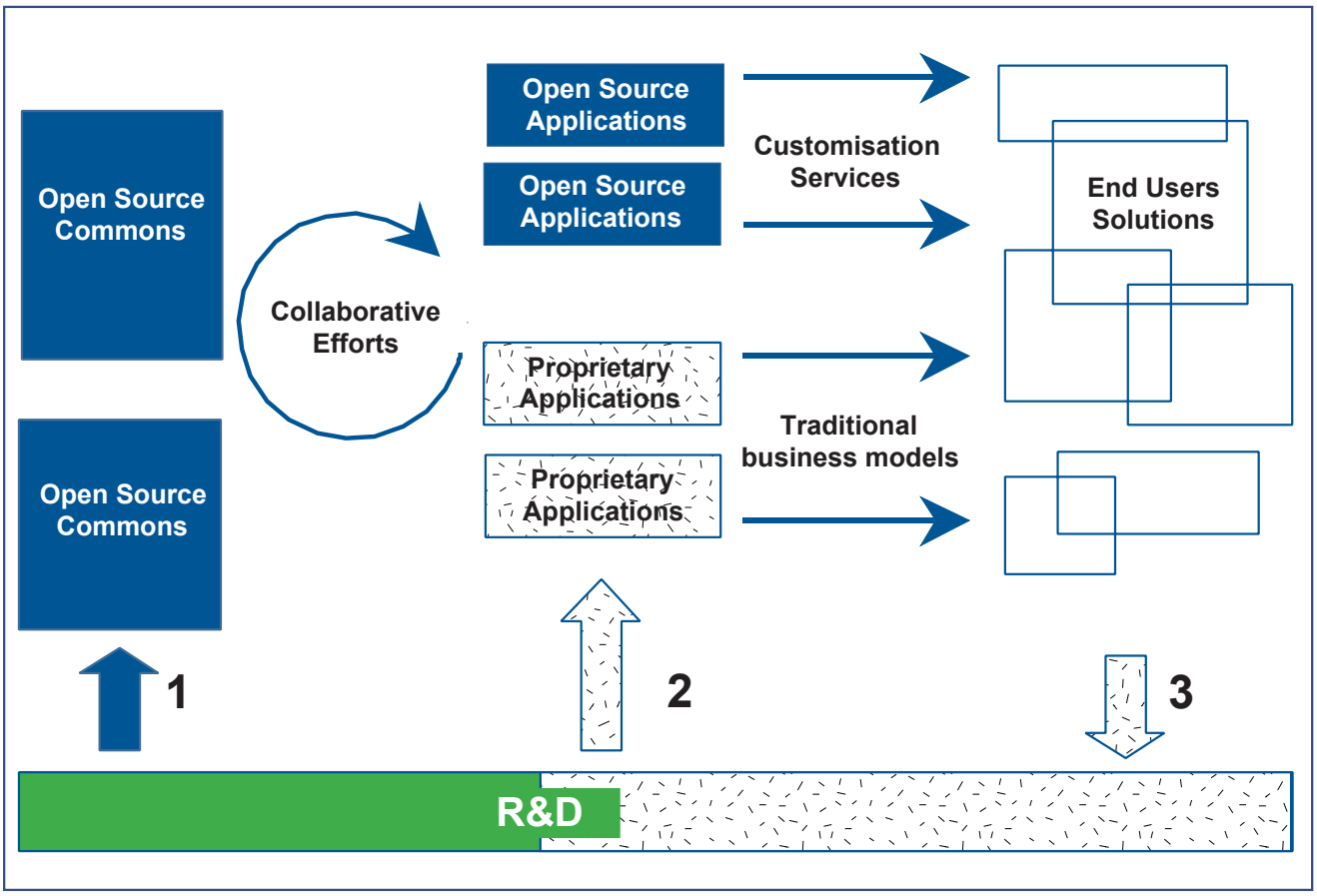


Figure 1 Possible R&D workflows

The effect of Open Source Software on this cycle is to give access to a wider R&D base (arrow 1) at minimum cost. It also enlarges the number of proprietary applications with a set of Open Source ones, allowing faster and easier ways to create standards, finally adding new systems, products and services to the repository.

Using open source software offers a number of advantages, such as the ability to reduce costs and

It is therefore recommended to **carefully study the license agreement and to make an assessment of the risks associated with these conditions**. One should always check whether one's own application is clearly separated from software under the GPL.

Open source software should also be screened for patent risks before use. The chance that a particular open source software package infringes on a patent is substantial.

Furthermore it is recommended to compare the open source software with one's own patent portfolio. If there is a chance that one's own patent is incorporated in an open source software package, then the use of that software package should be avoided. And, of course, compliance with the license conditions should always be checked.

By carefully applying the license conditions, it is possible to benefit most from using OSS while minimizing the risks.

2.5 Consequences

Open Source Software contains features that present European industries with both opportunities and challenges.

The opportunities stem from the fact that a huge, largely untapped source of innovation in universities and research centres makes a major contribution to the production of advanced software in a way that assures users of the quality of the work done.

This is in stark contrast with the usual way that software is marketed (through sales channels in which advertising is given as much emphasis as technical considerations). With OSS, considerable ingenuity is put into working out new de-facto open standards, which contribute to paving the way for renewed competition. Of equal interest is the reduced cost of acquisition of such pieces of software, since a large part of this cost (the associated R&D) is shared between the many holders of the rights.

There are also opportunities for European companies to jointly build or stimulate OSS as de-facto standards in areas where software infrastructure is sufficiently mature and has been standardized. This has two effects:

- it reduces the size of R&D investments, which have often become too expensive for companies to bear alone
- and it creates a standard that is not owned by any single company.

To get maximum benefit from OSS, European industry has to deal with a number of challenges. The first of these is to decide on the role OSS will play in their core business, i.e. make a conscious decision about the way they either will or will not use it. If they decide to use it for some of their activities, then they will have to adapt their organisation accordingly.

This is a relatively simple step compared to the more daunting one of changing the way they look at the whole software life cycle. "Make or buy" now becomes "make or buy or open source", which is a totally different way of looking at software development, since it means not only accepting but encouraging cooperation (possibly within a competitive context). This phenomenon, which is called "co-opetition" in ITEA's IRIS Book, requires a different approach to Intellectual Property Rights (IPR) that is strongly related to work on the software architectures of the products, systems or services provided.

Open Source Software may well be one of the best tools to escape (at least partially) from the monopolistic position that certain giant non-European companies have established in areas that are key for European development and independence.

In particular, it may also be one of the best tools for preserving and strengthening European access to and control of basic software for embedded systems in those application areas (e.g. automotive) where European software companies have a strong position, and where other global suppliers aim to extend their monopolistic positions elsewhere.

2.6 Conclusion

Depending on the business they are in, companies are likely to have different reasons for using OSS. In some cases, OSS can help them lower the cost of the service, system or product that they offer. In others, their contribution to OSS can help to establish new standards worldwide. By carefully applying license conditions it is certainly possible to derive considerable benefits from OSS, while minimizing the risks.

Open Source Software is not the "magic bullet" to solve Europe's software development competitiveness. However, OSS is an important new development and an interesting option for software-intensive systems. ITEA – as the premier European R&D Programme for software-intensive systems – will maintain its policy on OSS as to leave the decision to deal with OSS developments in ITEA projects to the project partners in the respective ITEA projects.

APPENDICES

A1 Technical aspects of OSS

A1.1 New development paradigm

A1.1.1 Change in the content of contributions from academia

A1.1.2 It's progress, but it's not necessarily THE solution

A1.2 OSS in the various technology categories of the ITEA roadmap

A1.3 OSS and quality

A1.3.1 What does quality mean for industry?

A1.3.2 What has been achieved so far with OSS?

A1.3.3 An industry perspective on quality

A1.3.4 Areas for improvement

A1.4 Research directions

A2 Economic and industrial aspects of OSS

A2.1 Software in the European Market: numbers and trends

A2.2 OSS and the software industry

A2.2.1 With OSS, free gets a value

A2.2.2 OSS allows shared R&D

A2.2.3 OSS enables reusability

A2.2.4 OSS reinforces sustainability

A2.2.5 OSS and business

A2.3 From utility to embedded software

A3 Legal implications of OSS

A3.1 Introduction

A3.2 Legal issues

A3.2.1 Copyright and licensing

A3.2.2 Liability

A3.2.3 Patents

A3.3 Risks when using Open Source Software

A3.3.1 The architectual level

A3.3.2 The business level (patent risks)

A3.4 Conclusion

A4 References

A5 Glossary

A6 Members of the ITEA Open Source Software Working Group

A1 Technical aspects of OSS

The aim of this chapter is to provide insight into the technical aspects, scope, process and quality of Open Source Software (OSS) as well as the extent to which it meets industrial requirements. These issues are examined in the light of existing material from a variety of available sources such as books and Internet publications, as well as discussions within the group of ITEA partner companies.

A1.1 New development paradigm

From the very early days, OSS development introduced a new development paradigm. First of all because it represents *collaborative work*. Even if an OSS project starts off individually, publishing the sources gives the author an opportunity to establish direct contact with users, who can report bugs and make proposals for improvements. If they have sufficient knowledge and ability, users can join the development team.

Development is, of course, *distributed*. The technical enabler and main vector of OSS projects is now the Internet. OSS project development workspaces such as Sourceforge.net and even basic tools such as CVS are designed for working remotely. CVS (Concurrent Versions System) – a widely used tool for source version management – is based on the concept of a repository located on a remote server. Sourceforge.net contains a set of tools (including CVS), which are hidden behind a web interface and thus easy to use from any operating system.

Of course, some big OSS projects such as GNU started years before the Internet was commonly available (around 1980), but they encountered considerable distribution problems compared with the LINUX project, which was started 10 years later and so could immediately make use of Internet tools and facilities. Internet is not an absolute prerequisite for starting an OSS project, but it does greatly facilitate the start-up and management of such a project.

A1.1.1 Change in the content of contributions from academia

Traditionally, most academic contributions to computer science have focused on purely theoretical aspects. However, the development of OSS gives those in academia the opportunity to start real projects, the results of which are directly usable by industry. Indeed, the best-known OSS projects started in an academic environment. In addition, OSS creates many opportunities for collaborative work between those in academia and researchers in private companies. It also allows universities to share valuable tools.

Below are some major examples of OSS projects that started off in an academic environment:

- BSD Unix from the University of Berkeley
- The GNU project started at MIT
- X Window System, which was started as the Athena project at MIT
- LINUX, which was started at the University of Helsinki.

Here are some additional OSS projects started by academics:

- Net-SNMP stack, which was started at Carnegie Mellon University
- The Second Extended Filesystem (ext2), which was started at the University of Paris (Jussieu)
- The RTAI (Real Time Application Interface) real time extension to the LINUX kernel, which was started at the Politecnico di Milano (Italy)

A1.1.2 It's progress, but it's not necessarily THE solution

Most open source concepts apply first and foremost to software, because most major open source contributors are involved in the software industry. However, this is not a good way to look at open source as a stand-alone strategy or as a business model (see the sections on business and legal aspects in Appendices 2 & 3).

The most important point to consider is the powerful influence of the open source model on intellectual property and therefore on software architecture. If one were to build a software solution with components such as LINUX, intellectual property rights and licensing would tend to separate:

- open source components that could, for example, exist in the kernel space (covered by a GPL license)
- closed source components that could exist in user space where a LGPL licence allows a mixture of proprietary components and LGPL components

Of course, using a completely different kind of license – such as Berkeley's – could lead to another kind of architecture being built.

What is of interest here is to build solutions based on OSS infrastructure components and to add solutions that might also be traded as conventional software. So it is crucial to find solutions where given OSS components can be integrated seamlessly with value-added elements that provide real benefits for customers. This is already happening in the world of embedded devices. Companies are able to build systems with greater functionality at a much faster pace when they can choose from a wide range of reliable OSS infrastructure components such as kernels and middleware, which allow them to focus on their specific application. Thus OSS has already become an enabler for larger numbers of products that are cheaper and sophisticated.

A1.2 OSS in the various technology categories of the ITEA roadmap

ITEA has defined several technology categories in the current roadmap and has projects in which OSS plays a role in all of these sub-categories. We can, in fact, rate them (from 0 to 5) to estimate the value of current OSS solutions compared with proprietary solutions. A value of 5 means that open source implementation should offer the best solution, whereas a score of 0 means there is a lack of support in the domain. OSS technologies focus on system services, as they usually originated in computer science laboratories (e.g. Berkeley University and MIT).

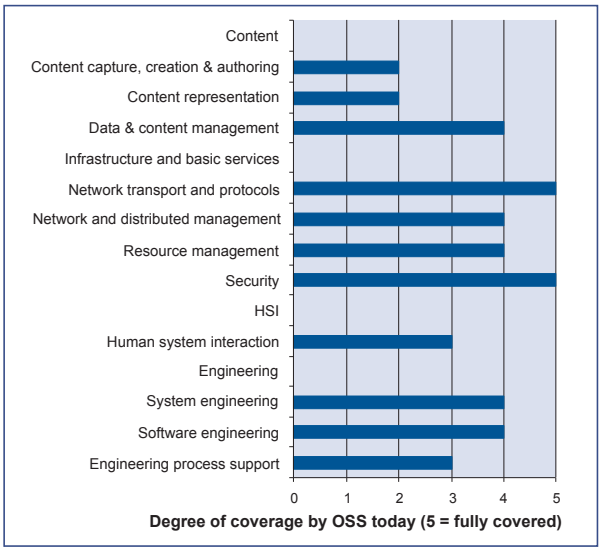


Figure A-1 Technologies in the ITEA Roadmap and corresponding OSS coverage

The bar chart in Figure A-1 shows the technological areas in the ITEA roadmap and the corresponding degree of coverage by open source software that is currently available.

Notes:

- **Content**
  Content does not provide the best example of OSS, except for data & content management, which is closer to system computing (clustering, high availability).

- **Infrastructures and basic services**
  This is a category where OSS is prominent, particularly in network protocols and security. Most well-known network protocols derive from OSS specifications and are defined in public RFCs (Requests For Comment). Many security technologies come from the OSS world (PGP, SSL).

- **HSI**
  Proprietary systems such as MacOS and MS Windows are renowned for the quality of their human interface. This is not yet true for OSS, but one of the best-known human interfaces (X Window System) is also a major OSS project. OSS projects such as GNOME /Gtk or KDE/Qt are currently close to achieving an adequate standard.

- **Engineering**
  The open source world provides component-to-design OSS, but sometimes it cannot be adapted to design proprietary software because the methodologies are very different. Usually choosing an OSS strategy means using OSS tools – most of which are very good – to design software, but the developers need to be dedicated to the OSS methodology and culture (for example, attempting to develop a program based on Windows MFC using OSS tools may not be the best approach).

A1.3 OSS and quality

A1.3.1 What does quality mean for industry?

For industry the term quality is strongly related to issues such as safety, reliability, availability, maintainability, testability, portability, security,

process maturity and conformity to certain standards. Thus in many cases quality deals with non-functional requirements. A widely accepted understanding of quality in industry is that quality often relates to the extent to which a product and/or process conforms to a given set of requirements.

The issue of quality is a topic that's hotly debated. Influenced by the discussions that took place in European industry during the 1980s, which was a response to the Japanese 'quality revolution' at that time, a number of general concepts emerged. According to Garvin [Garvin 1984], there are various points of view regarding quality: a transcendental view, a user view, a manufacturing view, a product view, and a value-based view. In terms of quality related to the industrial development and use of OSS, the manufacturing view and the product view are the most relevant.

The manufacturing view describes the degree to which a software packages satisfies formal requirements. This is, in fact, the main focus in software testing. According to the manufacturing view, quality is defined in terms of numbers of defects and other metrics. The product view of quality is closely linked to the inherent properties of the software once released.

A1.3.2 What has been achieved so far with OSS?

The OSS development process is meanwhile widely considered an effective method for reducing cycle-time and cutting costs for design, implementation, and quality assurance [Feller Fitzgerald 2002]. This is especially true for operating systems such as Linux and system infrastructure software, e.g. web servers, compilers, editors, middleware, print systems and file servers. In this realm, when talking about the benefits, almost all authors writing about OSS emphasize the high quality of OSS as a key advantage [Schmidt Porter 2001]. Whether it is sufficient for industrial grade requirements needs to be determined.

In contrast to the classical software development route, in the OSS development model code is released early and often. As Raymond [Raymond 2001] puts it, the code must run before you release it to the community and you must be eager to accept improvements and changes to the code. This way the community helps with debugging and re-factoring in a way that was not common before the advent of OSS. On the other hand, there are also commercial projects that successfully follow a similar development method. However, the involvement of the user community is different. Given full access to the source code an active community of developers and users is able to use, check and debug a program in a parallel mode that we have not seen before. Also the authors get feedback that is much more helpful than, for instance, that provided by users of Closed Source Software, CSS.

There is evidence that the open source development model has the capability to match and even overtake commercial counterparts when it comes to software quality. A study carried out by Reasoning Inc. showed that the LINUX TCP/IP stack (2.4.19) is the best in its class with 0.1 defects per KLOC compared to commercial stacks with an average of 0.55 defects per KLOC [Reasoning 2003]. Of course one cannot extrapolate these statistics to other areas of Linux (e.g. drivers) or to OSS in general.

The only thing that cannot be done with software quality in an OSS process is that it cannot be enforced. It just happens intrinsically as a natural consequence of the underlying terms of operation of the OSS process itself. Thus from a classical point of view there is less control over the participants involved in order to enforce adherence to a certain process.

There is a clear difference between the way quality is achieved in conventional processes and in OSS processes. Regarding the above-mentioned distinction between the manufacturing view and the product view, it is necessary to relate the manufacturing view to the software development process and to map the product view to the extent to which the final software solution is degree error-free.

Conventional or industrial processes move through a series of phases: requirements, design, implementation and testing, and are accompanied by appropriate documentation and quality assurance procedures. OSS, by contrast, starts with a core of ideas and some working code, the so-called working

prototype. After that, given a sufficient number of co-workers, the software grows in functionality and code quality. There are examples of high-quality results, e.g. Linux. But, on the other hand, there are also examples of bad quality. What's more, there is a strong relationship between the number of users and/or the degree of use of that software and its perceived or real quality.

A1.3.3 An industry perspective on quality

Many of the industrial stakeholders of ITEA are developing or implementing software-intensive systems in industrial environments (such as industrial automation, telecom, power infrastructure, automotive systems, aircraft and space systems). Dependability has always been essential when building and maintaining such systems. The more the systems are used the greater the impact of software quality.

For certain areas of industry, especially safety and mission-critical systems, the quality of the software in a given system must be demonstrable. Different application domains demand different levels of quality, which must be assessable with reliable metrics and thresholds. In addition to having a manageable process for the software development itself, a project building dependable systems must incorporate rigorous and traceable documentation management. The approach in industrial projects and in OSS projects is different, especially with regard to documentation.

When it comes to the use of OSS in industrial grade projects, quality issues related to 'imported' components are always critical. The question is how to obtain the level of quality for the envisioned software component that fits the overall quality goals of the project. For example, the level of maturity of the Linux kernel might be acceptable, but how do you assess the quality of drivers that may very well influence the stability of a certain system? Some kind of inspection and improvement measures needs to be taken for the software incorporated in critical application environments.

A1.3.4 Areas for improvement

In light of the above, quality issues need not be a problem per se. In fact, OSS often sets a new standard for software quality compared with CSS products for certain types of 'normal' software-intensive systems. Thus, for a wide range of applications, OSS can also compete with traditional software in terms of quality.

However, concerning characteristics of dependable systems such as high integrity, safety and mission-critical systems, more elaboration is needed. When systems in this category are to be equipped with open source software, additional measures are required. The following list can be used as a guide through issues relevant to narrowing the gap between the status quo and a possible scenario with higher quality systems:

- Representatives of industry could act as quality coaches for certain open source components that might be used in dependable systems, which may also include aspects related to quality assessments, improvements and certification.
- A better exchange of ideas on the key success factors related to quality could bring about a win-win situation between the OSS community and industry partners, where industry comes to an even better understanding of the success factors generated by OSS's quality and the OSS community can benefit from rigorous quality approaches in industry, which should also include documentation standards.
- Code analysis: what can be done with tools should be done. Open Source Quality (OSQ) aspects related to formal aspects of software construction, type safety, formal verification of state machines and automatic program verification have already been addressed by the OSQ group at Berkley [OSQ 2003]. The Hayards project of Sun, Parasoft and others also addresses better issues such as improvements in testability, currently mainly for Java code [StreamingMedia 2002].
- Providing enough test space, investing in rigorous quality checks, e.g. continuous testing.
- Promotion of industrial grade quality and performance assessment to gain insight into many more qualities of OSS.
- The use of OSS in safety-critical applications is not yet solved, at least not in industry[A1&A2]. Some key European projects are likely to lead the way in this area.
- Software distributors and integrators who build customer solutions comprising a mixture of open

[A1] Réseau d'Ingénierie de la Sûreté de fonctionnement» with contributions from: Airbus, Astrium, LAAS-CNRS, Technicatome, Thales, ESA, SNCF, LSV, INRETS, IRISA) http://www.ris.prd.fr/GT/LL/JourneeGT-LL.htm

[A2] Upon request of the UK Health & Safety Executive (HSE), the UK MoD and the Safety Regulation Group of the UK Civil Aviation Authority: "Preliminary assessment of Linux for safety related systems".http://www.hse.gov.uk/research/rrhtm/rr011.htm

and closed source software could be motivated to offer some type of quality clearance for the OSS software they use and redistribute.

The above items are summarized in A - D in Section 3.3.

A1.4 Research directions

One big advantage of open source is that it is freely accessible. So, any sophisticated tools that become available can be applied immediately to whatever software needs to be checked. Researchers can choose from a huge selection of real-life examples that are not available in a CSS environment. Having quality assessment tools at hand, users are able to deal with software quality – if they wish at a very basic level, namely at the level of the code-base itself. Different test approaches might reveal different deficiencies in the code under inspection. The combination of these different approaches can increase the coverage of improvement measures for the applied software package.

An interesting idea might be to take the quality aspect of OSS as a subject for software engineering education within the participating countries. Thus it should be possible for a large audience to come to understand the basic principles and advantages of open source software development and its relation with quality, relatively quickly. But it would also be possible to address the other areas discussed here such as industries needing advancement through software development[A3].

[A3] Workshops on Open Source Software Engineering", from "International Conferences on Software Engineering (ICSE)" since 2001. http://opensource.ucc.ie/

A2 Economic and industrial aspects of OSS

Open Source Software[A4] has brought about a significant change in the way software is produced and used, altering both technical and economic traditions in a quite disruptive manner. Making profit out of something that is essentially free is a highly innovative idea. Establishing Intellectual Property Rights (IPR) based on something public is even more revolutionary. Generally what is free has no value and what is open cannot be closed. The aim of this chapter is to describe and illustrate what Open Source Software represents in terms of an economic business model.

There are two main categories of software. One is now considered a utility by most businesses, much like electricity or communication. It encompasses "basic software" (e.g. Operating Systems) or database software or software for the management of businesses (from procurement to electronic mail, through server control, Computer Aided Design, etc). The other is the power behind the capabilities of advanced appliances, ranging from an aircraft to a refrigerator (so-called "embedded" software).

It is in the first type of software that OSS has been making greater inroads in recent years. It is also the domain in which most data are available and clearer trends are becoming visible. However, for Europe, the second type of software is just as important, since it controls applications in which European industries are world leaders such as automotive, railways, telecommunications, consumer electronics and aeronautics. It is also the software that will, in the near future, enable "Ambient Intelligence."

Our approach in this chapter will be to analyse the first category and try to see if it is possible to draw lessons for the second one by addressing the following subjects:

- A brief description of the current state of software in the European market
- A short analysis of the impact of OSS on this market and of the reasons why it is making inroads
- A discussion of the potential of OSS for system integrators and more specifically for makers of embedded software.

A.2.1 Software in the European Market: numbers and trends.

A recent survey in Europe by Audoin [Audoin 2002] shows a € 31 billion market for 2002 and predicts up to € 45 billion in 2006, which represents 9.1% growth (*see Figure A-2*). It is clear that the leading suppliers in the European Market are based in the US.

The Western European market for application software products and solutions by country - 2000-2006 - Million Euro

	2000	2001	2002	2003	2004	2005	2006	00/01	01/02	AAGR 02/06
Austria	607	640	661	692	751	827	906	5.4%	3.3%	8.2%
Belgium	530	570	595	638	708	791	878	7.5%	4.4%	10.2%
Switzerland	1,198	1,289	1,347	1,472	1,627	1,787	1,951	7.6%	4.5%	9.7%
Germany	8,476	9,180	8,973	9,347	10,343	11,448	12,548	8.3%	-2.2%	8.7%
Spain	1,070	1,196	1,266	1,369	1,501	1,663	1,844	11.8%	5.9%	9.9%
France	4,848	5,304	5,130	5,361	5,948	6,655	7,335	9.4%	-3.3%	9.4%
Italy	2,579	2,855	3,019	3,269	3,661	4,110	4,551	10.7%	5.7%	10.8%
Netherlands	1,210	1,290	1,220	1,270	1,410	1,585	1,760	6.6%	- 5.4%	9.6%
Nordic	2,800	2,950	2,906	2,963	3,249	3,613	3,974	5.4%	- 1.5%	8.1%
Portugal	347	395	403	425	473	541	620	13.9%	2.0%	11.4%
United Kingdom	5,049	5,266	5,365	5,636	6,163	6,747	7,299	4.3%	1.9%	8.0%
others	509	559	573	605	680	761	839	9.8%	2.5%	10.0%
Western Europe	**29,223**	**31,494**	**31,458**	**33,047**	**36,515**	**40,527**	**44,505**	**7.8%**	**- 0.1%**	**9.1%**

Figure A-2 The market for application software in Europe. *© PAC (Pierre Audin Consulting)*

[A4] OSS represents an original way to use copyright laws and to enhance the value of derived work. Compared to proprietary software, these laws are used in reverse with OSS. Instead of restricting access to code, the owner of the copyright gives everyone the right to use, modify and distribute it, while fully respecting Intellectual Property and allowing Industrial Property to be shared.

Leading suppliers of application software products and solutions to the European market in 2001 in Million Euro							
Rank	**Company**	**Nat.**	**Rev.**	**Rank**	**Company**	**Nat.**	**Rev.**
1	Microsoft	US	3550	20	I2	US	165
2	SAP	D	2315	21	EDB Business Partner	NO	150
3	IBM	US	550	22	Hewlett-Packard	US	150
4	Siebel	US	490	23	Mentor	US	145
5	Oracle	US	340	24	Computer Associates	US	140
6	Sage	UK	335	25	Alcatel	F	130
7	Autodesk	US	305	26	Amdocs	US	130
8	Cadence	US	295	27	Invensys	UK	130
9	Dassault système	F	295	28	NCR	US	130
10	Adobe	US	274	29	Exact	NL	129
11	CSC	US	260	30	TietoEnator	FIN	125
12	PTC	US	260	31	Wincor Nixdorf	D	123
13	PeopleSoft	US	215	32	Unisys	US	120
14	SchlumbergerSema	F	215	33	Intentia	SWE	115
15	Datev	D	205	34	Torex	UK	115
16	EDS	US	200	35	Getronics	NL	110
17	Misys	DK	193	36	RM plc	UK	105
18	Navision	DK	175	37	Netmetschek	D	100
19	CMG	UK/NL	165	38	synopsys	US	100

Figure A-3 Leading suppliers of application software to the European market *© PAC (Pierre Audin Consulting)*

There is only one significant European challenger: SAP (*see Figure A-3*). This figure also shows the predominance of major suppliers such as Microsoft, IBM, Siebel and Oracle. US suppliers get € 7.8 billion (of the € 13 billion revenue generated by 38 listed companies), i.e. 60%. Microsoft, IBM, Siebel and Oracle with combined revenues of € 4.9 billion, representing 38%. From a financial point of view, the market capitalisation of these leaders is impressive: e.g. $272 billion for Microsoft and $127.5 billion for IBM in 2002. It's clearly not going to be an easy task for European software suppliers to challenge such giants.

A new unknown player called Open Source Software has appeared on this market in recent years. A leading type of Open Source Software Linux, which is a Unix clone, is currently used on approximately 16% of servers worldwide (*see Figure A-4*).

Another successful example of Open Source Software is Apache – a free piece of software that is 100% compliant with web standards.

Worldwide Server Market 2002 by Operating System Platform				
OS	**2000**	**2001**	**2002**	**2003**
OS/390	0.1%	0.05%	0.05%	0.04%
Unix	18.0%	15.4%	14.5%	13.9%
Windows	54.3%	59.5%	60.4%	60.5%
Other	2.3%	1.7%	1.3%	1.1%
OS/400	0.9%	0.6%	0.5%	0.5%
Linux	10.1%	11.4%	13.3%	15.9%
Novell	14.4%	11.4%	9.8%	8.1%

Figure A-4 IDC, March 2003

These days Apache runs on any platform (from Linux to other Unix systems and Windows) and is used on over 60% of Internet web servers (*see Figure A-5*). Apache is now a "*de facto*" standard for web servers and it is increasingly difficult for competing software to challenge it.

The US Administration is seriously considering using OSS. A recent conference on "*Open Standards/ Open Source for National and Local e-Government*

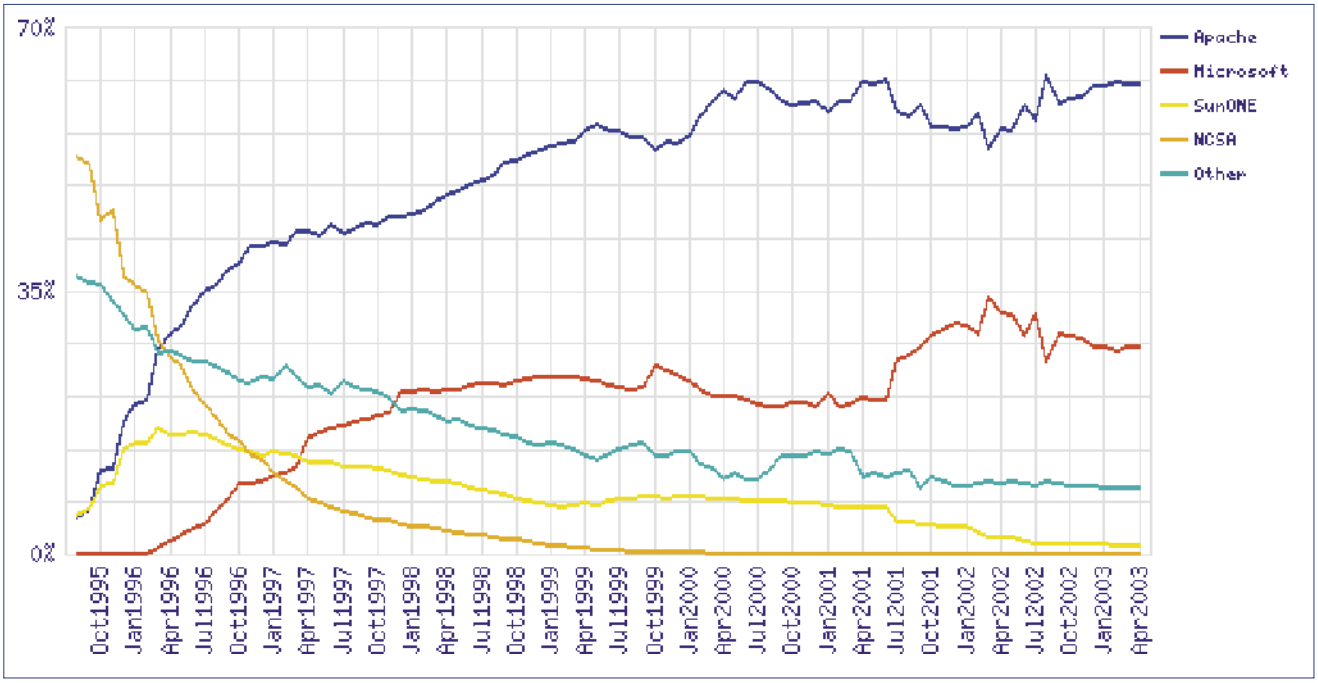


Figure A-5 Netcraft April 2003 Survey

Programs in the U.S. and EU" held in Washington, DC, to promote best practices, raised awareness and allowed policy makers from the U.S. and Europe to share experiences[A5]. MITRE is a long-time sponsor of Open Source Software and recommends its customers to take a close look at it [Kenwood 2001]. According to MITRE, "*Free Open Source Software plays a more critical role in the DoD than has generally been recognised"*.

For the US software industry, OSS represents such a significant trend that major players are actively considering collaborating on Open Source Software production. Large US companies such as IBM, HP and Sun have major Open Source activities[A6], which put them in a very strong position to get the maximum benefits out of OSS.

Europe is also very active in OSS development. Various statistics show that around 70% of OSS developers live in EU countries (compared with 13% in North America). Figure A-6 shows the extent of Linux use in the early days. But there's an important issue here. On the one hand Europe is well represented in the community of OSS developers, while on the other there is a risk that this development will be dominated (and therefore controlled) by US companies. FLOSS[A7] highlights this point: "As large companies such as IBM, HP and Sun are US-based, it might be expected that concerted support for open source development would be more US-centric. As a result, a peculiar situation has arisen in which – although the majority of developers are European – key decisions end up being made in the US."

OSS is also very attractive for the European market. In the case of Linux, IDC reports 170,000 servers using Linux (sold by hardware vendors) in Europe for 2002 and predicts sales of 670,000 for 2006. According to FLOSS, 27% of companies and public institutions are currently using OSS in Europe. This report strongly recommends the use of OSS in Europe.

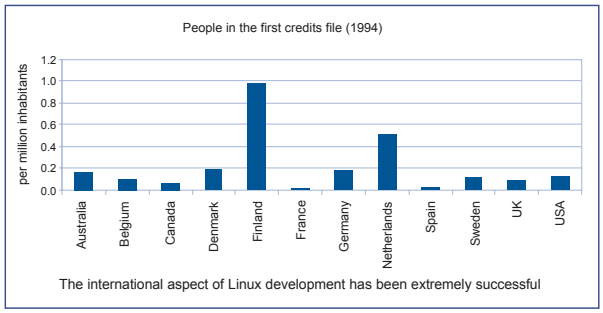


Figure A-6 The Political Economy of Open Source Software (Steven Weber, BRIE, University of California, Berkeley: Linux developers)

[A5] http://www.egovos.org/

[A6] http://www.berlecon.de/studien/downloads/200207FLOSS_Activities.pdf

[A7] European Commission's Free/Libre an Open Source: Survey and Study

Concerned by the stakes and challenges of e-government, public organisations in Europe are very proactive in using Open Source Software nowadays encouraging shared efforts e.g. in Germany[A8] and in the UK[A9]. The justification for such policies lies in the need to come up with solutions that provide value for money, to ensure the interoperability of governmental systems, to reduce cost and risks, and to guarantee the security of vital information systems.

Despite the quasi-monopolistic position of some products such as MS Windows or Oracle's DBMS and without any major company driving its production, Open Source Software has nevertheless gained significant momentum. This "*new player*" is now seen as a serious challenger to top-rank companies[A10].

Basically Open Source provides standards Software tools in domains where Software penetrates deeper everyday. This trend to converge is true for almost any domain and it is fundamental in Industry's roadmap. Establishing standards represents a long-term effort for large companies in a competitive environment. It is almost impossible for small players. Examples such as Linux or Apache show that "*Open Source Software is almost automatically the source of de facto standards for any number of protocols or systems" (FLOSS)*. Long time before Linux or Apache, TCP/IP software such as "bind" was Open Source. As Internet is the key success factor of OSS both are strongly connected in a virtuous circle.

A2.2 OSS and the software industry

In the mid-1980s a major software vendor such as Microsoft preferred proprietary protocols to TCP/IP[A11] for obvious reasons: Open Standards do not tie a customer to a vendor. But with the massive adoption of Internet, Microsoft rapidly adapted its strategy, proposing TCP/IP as the default network protocol for all of its products, because standardisation helps to increase the size of the market. Thanks to the standard open Internet protocol, Microsoft has had the opportunity to communicate with a larger number of customers. Let's now look see deeper at why Open Standard and Open Source have attracted so much interest within the software industry.

A2.2.1 With OSS, free gets a value

When we analyse the reasons for the success of a disruptive technology such as the Internet, the fact that most of the underlying software is available at no cost has certainly been a good way for American companies supported by the US government to gain significant market share. Massive investments based on TCP/IP protocols were very helpful in developing the US computer industry's dominance, but also in globalising the telecom industry – thanks to digital networks. The low cost of adoption sustained this investment. "*Free*" in this case provides a strong incentive to use the "*free*" software. The more people use it, the more the software becomes useful. These indirect network externalities rapidly created a "*de facto*" standardisation. But "*free*" is just one of the reasons. Open Standards and Open Source implementation of these standards explain the rest.

Having access to code specifications (Requests for Comments) and the code itself (the free software stack of TCP/IP) created a virtuous circle around the Internet comparable to the principle of publication within the scientific research community. It enabled rapid technology propagation, providing free access to knowledge contained within the source code. Since those days, this story has been repeated with web technologies (html standards and the http protocol from CERN) and today with middleware layers such as web services (from XML/RPC to SOAP and UDDI and WSDL). Standardisation is a major battlefield for technological competitors and OSS provides an efficient way to establish "*de facto*" standards.

A2.2.2 OSS allows shared R&D

The recent industrial interest in Open Source is mainly due to the ability to gain access to shared R&D. The Open Source Community is basically a set of loosely coupled individuals, some of whom may be paid by companies. These individuals have common technical interests. They share the development of software and the results of such development. Thanks to the licensing mode used, no one is reluctant to share work because the result is not exploited by anyone in particular and everyone can build business around it. Shared R&D enables development at lowest cost, using a common software base. It enables companies to focus on aspects that are more valuable for their own business. This is illustrated in a MITRE report

[A8] http://linux.kbst.bund.de/

[A9] http://www.e-envoy.gov.uk/oee/oee.nsf/sections/frameworks-oss-policy/$file/oss-policy.htm

[A10] This is described in the International Herald Tribune [IHT 2003]. Microsoft's reaction clearly shows that it is feeling the heat.

[A11] TCP/IP was not the only protocol on the field at the time; ISO, for instance, was another important competitor.

describing Linux as a collaborative effort of 120,000 volunteers, representing $2 billion worth of labour. This effort has already generated more than $2.1 billion in revenues for hardware vendors in Europe (figures from 1999 to 2001 - IDC). Sharing R&D has other advantages, for example, it enables knowledge sharing and the education of skilled people with a minimum of effort compared to traditional courses.

A2.2.3 OSS enables reusability

Another important result obtained from OSS is **reusability**. For years, reusability has been the software industry's "*Holy Grail*." Reusability is traditionally seen as costly technology with a mid to long-term ROI. OSS drastically changed this perception because it represents one of the first tangible successes of low cost reuse of software. The OSS world can be seen as the largest library of reusable and adaptable components. Symptomatically the French ADAE, which is in charge of promoting best practices for e-government within the French Administration, clearly associates "*Free Software*" and "*Software Reuse*" in its publications[A12]. The use of Linux and other Open Source components within embedded devices provides a good example to industry. As MITRE puts "*The modularity of Linux enables it to be used in a wide range of systems, from a supercomputer to a refrigerator… Margins are very low in embedded devices. The free cost of Linux helps this market. Using open source for embedded systems avoids the licensing fee from closed source vendors, which amounts to a large cost savings for manufacturers producing large volumes of embedded systems. Developers of embedded systems can compress Linux and tweak it to suit the needs of their specific applications. Developers world-wide can co-operatively enhance the software and fix bugs in real time*."

A2.2.4 OSS reinforces sustainability

The sustainability of components is also critical for the companies that use them. MITRE also explains that the sustainability of OSS is due to the Open Source mode of licensing: "*The lifetime of open source licensed systems can be extended indefinitely since the source code and documentation are freely available. Information requiring long-term access will not become buried in obsolete or non-functional, undocumented, proprietary formats every few years. With open source licensing, the user can develop the software in-house, outsource to the original vendor, or outsource to an after-market support vendo*r."

A2.2.5 OSS and business

In [Meta Group EO 2003], Meta Group explains pragmatically how OSS is good for both innovation and business. This document emphasises the fact that "*seeking to address least-common-denominator feature functionality, open source software is unlikely to lead innovation, but through 2007, users can expect to see vendors drive innovation by combining open source software and commodity hardware to address critical business issues at lower costs. Commercial vendors will continue to respond to the open source challenge by offering low-cost entry versions of products (e.g., OS, database, middleware) and up-selling to their "enterprise" feature/function-rich versions for organisations that desire more than "good enough" software*." Business and OSS are not incompatible. When well-used, OSS should increase revenues , for example by reducing R&D costs and accelerating time-to-market. OSS provides common and "*business-neutral*" building blocks to entrepreneurs who can develop a business based on them in a completely independent manner. It is up to them to focus development on a specific value proposition that they have identified as being able to generate profits.

OSS also opens up opportunities for vendors of professional services. Because of access to the code and the need for OSS to be customised or integrated to build a valuable solution, money that is not forth-coming for proprietary licences will switch from ISVs to SIs. Because the OSS world is huge, SIs mainly sell confidence by providing expertise and professional services (integration, installation & support, certification, training, upgrade services and bespoke software) on top of OSS. One very valuable benefit that OSS brings to SI is access to a vast community of expertise and the kind of access to R&D that is simply not viable within a classical SI business model.

Meta Group presents an interesting vision from a software vendor's perspective. In [Meta Group MySQL 2003], MySQL is described as "*the wildcard in the competitive $12 billion database market*." It explains how the Swedish company MySQL AB "*is attacking*

[A12] http://www.atica.pm.gouv.fr/pages/documents/liste_liaison_theme.php?id_chapitre=8&id_theme=55&ancre=debut

market leaders using a combination of open source and commercial licensing to speed the adoption of its software … MySQL AB's use of the open source software model is propelling its adoption at a much faster rate than any traditional software sales model could have done. IT organisations should become familiar with this new software model, because it will continue to impact infrastructure buying options in the future." The conclusion of this report is that "*OSS models promise to significantly reduce software expenditures and thus effect a positive change in the client/vendor relationship*".

A2.3 From utility to embedded software

The previous section describes how OSS came to play a major role in utility/commodity software. It is also making inroads in some embedded environment such as tablets, PDAs, set-top boxes and industrial automation[A13]. However, some stakeholders in these domains are not yet convinced of the value of OSS for their businesses. There are two main reasons for this, reflecting a variety of technical, legal and business issues.

The first reason is related to the fact that most of the value brought by OSS depends on the size of the community who shares the development and participates in the life cycle. While the numbers are high for sharing common utilities such as a database system or an OS, some applications lack sufficient numbers of developers. The more the application diverges from utility software, the smaller the associated part of the OSS community. To benefit from the "OSS effect", these businesses should certainly team up with others sharing either the same kind of problems (e.g. the need for extremely reliable, resilient real-time systems) and/or the same architecture. This would enable them to go beyond the benefits of simply re-using an OS or a server-enabling piece of software towards defining and implementing actual platforms and/or frameworks that could easily become *de facto* standards thanks to their OSS status.

Another factor influencing the extent of the support for OSS is that most of the software professionals working on it come from academia. This means that they are always very interested in developing software solutions for novel problems or those of scientific interest. Unfortunately, there is a need in industry to solve more modest but equally vital problems, e.g. a "quality software workbench" that would help would-be users assess features, function, performance and quality of the OSS components they would like to use. In many mission-critical applications, the lack of such facilities clearly hinders the use of OSS.

The second reason is related to the fact that many organisations are unfamiliar with the way the OSS community works and are therefore afraid of losing their intellectual/industrial property or of having to depend on unreliable "suppliers" if they use OSS components. This is an important issue that requires a change of perspective before it can be solved. In fact, if OSS is used simply as an opportunistic, one-off application, there are indeed more risks than opportunities. The table in Figure A-7 outlines some of the business implications of OSS, taking a diagnostic approach from the viewpoint of a systems integrator (not that of a software editor nor a computer manufacturer nor a company providing services, whose concerns are different).

To benefit from opportunities while minimizing and controlling threats, a long-term relationship must be built with the Open Source movement. This means that one should mainly focus on organisational issues, the most relevant of which are listed below:

- evaluate technical aspects (functionality, performance, quality)
- investigate OSS life (development model & management, communities, support)
- investigate external support and expertise related to the OSS selected
- survey updates of the OSS selected
- survey trends
- manage making versions of the selected OSS (with the aim of maximizing stability)
- manage knowledge gained (KM)
- manage awareness related to OSS risks (licensing issue) and the existence of the processes listed above.

Some of these issues are common to a large number of integrators and more generally to those working in the software industry, so they could be addressed through mechanisms similar to those used by ITEA.

[A13] http://www.linuxdevices.com/

OPPORTUNITIES (Explanations in green)	THREATS (Explanations in green)
To preserve the strengths of Off-the-shelf components (COTS) with equivalent: • Time to market • Cost to market • Technology to market OSS and COTS are external component that may reduce R&D efforts	External components Output of an external process we don't yet know much about (design, coding quality, developers, testing, etc)
… and even improve on them Technology is available sooner, it matures sooner (thanks to the responsive interactions among communities), and acquisition & deployment are generally free of charge	Licensing issues We risk losing IPR
To enforce system quality Adapting OSS makes it possible to match target systems' requirements exactly (all of them and only them); no turn-round time is needed to deal with COTS bugs or add missing elements; and there's an opportunity to clean up 'dead' code	Evolution and release frequency Evolution can get out of control Release schedules can be very fast
To leverage technical skills (and therefore corporate knowledge) Free access to and interaction with OSS R&D assets (source code, documentation, developers)	
To be able to take back control of systems Developers are no longer tied to the release policy dictates of editors; of course there remains a trade-off between stability and the evolution of technology	
To carry out sustainable investments OSS promotes (implements or creates) open standards Sustainability of OSS Acquisition of skills	

Figure A-7 Opportunities and threats presented by the use of OSS

A3 Legal implications of OSS

A3.1 Introduction

Open source software is a collective name for software for which the source code is freely available. Anyone is allowed to extend or improve the software and to distribute it. The conditions in licenses for open source software may in some cases create problems. One might for instance be required to release the source code for a commercial product if it is based on open source software. And patents related to the product may become worthless..

A3.2 Legal issues

From a legal point of view, there are three main issues: copyright & licensing, liability and patents. We briefly examine each of these below.

A3.2.1 Copyright and licensing

The author of a computer program automatically holds the copyright. This allows authors to restrict copying, use and modification of programs by third parties. To permit others to use the software, the authors have to give permission for whatever they want others to do with it. This is called a license. In the license the author sets out permissions and the conditions under which these are granted.

For example, the license for Microsoft Windows gives the user permission to install the software on one PC and has as one of its conditions that the user shall not hold Microsoft responsible for any damage that may result from using Windows. The license does not grant permission for copying or redistributing the software and the license is of course only granted if the user pays a fee.

- **Open source software**
  The approach taken with open source software is totally different. The software is freely available for anyone, including the source code. The license grants everyone permission to adapt or improve the source code, for example, to fix errors, to make a more efficient implementation or to add completely new functionality. The software may also be copied and distributed freely, even in modified form. The author does not charge for this. The software is freely available on the Internet. It is even permissible to charge for distributing somebody else's open software without having to share any of the revenue with the original author.

- **Open source software licenses**
  Like all software, open source software is protected by copyright. Although the license does not demand any money, this does not mean that there are no conditions. Almost all open source licenses require users to reproduce the name of the author (typically in the form of a copyright notice) whenever they distribute the work or incorporate it in a commercial product. Other licenses require a distributor to make the source code of the open source software available, for example, by shipping it together with their own product or by putting it on a website. It is almost always required to clearly identify modifications. Sometimes these modifications must be supplied in separate files, and modifications sometimes must bear a different name to avoid any confusion with the official version.

  The best-known open source licenses are the BSD license, the GNU General Public License (GPL), the GNU Library or Lesser General Public License (LGPL) and the Mozilla Public License (MPL).

- **BSD license**
  The BSD license is a very simple, broad license, which permits unlimited use of the software, as well as distribution in source code and object code form. The software can be adapted freely and be incorporated in other programs without any restrictions. The only obligations are that the copyright notice and a warranty disclaimer must be reproduced in the source code and in any documentation. The name of the author or that of any contributor may not be used to endorse or promote products derived from such software without specific prior written permission.

- **The GNU General Public License**
  The GNU General Public License (GPL) is the best-known open source license. Using, copying and distributing software under the GPL is always permitted. Modifying the software, or even developing an application using for example, a library under the GPL is no problem.

Distribution of a work that is *based* on software under the GPL is also permissible if the work based on this software is also distributed under the GPL. It is not allowed to impose any further restrictions on the rights under the GPL. The source code of the software as a whole must be made freely available. The recipient of the software has the right to freely use the entire package, to adapt and modify it, and to distribute it further without any payment to the author. Recipients may also distribute the software in modified form, as long as they do so under the GPL as well. The precise meaning of "based on" is discussed below.

There is no obligation to release modifications or extensions to software under the GPL. Private modifications, including modifications carried out internally within a company, can be kept secret. The obligation to make the modifications or extensions available under the GPL only arises if the person who modified the software chooses to distribute the result.

- **The Library General Public License**
  The Library General Public License (LGPL) is a variant of the GPL. As the name indicates, it is mainly intended for libraries (such as DLLs), programs with functionality that can be used by other programs. The LGPL permits linking the library with another program without imposing any obligations on that other program. Modifications to the library itself can only be distributed under the terms of the LGPL, which in this respect are the same as the terms of the GPL. That is, such modifications can only be distributed in source code form and without imposing any further restrictions on the recipients of these modifications.

- **The Mozilla Public License**
  According to the Mozilla Public License (MPL) the author of a work grants everyone a worldwide royalty-free license to use and distribute work under existing patents and copyrights. Anyone distributing modifications or other contributions to the work must grant the same permission for contributions and for the combination of the original work and such modifications.
  It is only permitted to distribute the (original or modified) work if the source code is also made available. One way to satisfy this requirement is to distribute the source code together with the object code. Another way is to put the source code on a website. Integration of software under the MPL with one's own application is permitted, as long as the portion under the MPL is made available including source code.

A3.2.2 Liability

Using COTS software the liability issue is addressed through the contract between the supplier of the software and the "user" who incorporates this software in his or her product. With OSS there is however, no formal or legal supplier and no such contract and it is therefore up to the user to bear the full liability, also of the OSS component.

A3.2.3 Patents

Some OSS licences and patents are essentially incompatible. The basis of the open source software development model is the sharing of source code and the right to use other people's code in one's own work. If that code infringes on a patent, distribution and use is not allowed without permission from the patent holder.

During the last ten years there has been a large increase in the number of patents issued for software-related inventions. Because the criteria for novelty and ingenuity are a lot easier to fulfil than many programmers expect, the general opinion regarding patents in the open source community is rather negative. The term "software patent" appears to be almost synonymous with "something trivial." And many in the open source community, especially in Europe, consider software patents to be invalid by definition, because the European Patent Convention excludes patents on computer programs as such[A14]. Current case law of the Board of Appeals of the European Patent Office, which permits patents on software if there is a "technical effect", is usually ignored or considered to be contrary to the spirit of the European Patent Convention by the OSS community.

Should copyright problems arise, it is always possible to replace an infringing program with an independent

[A14] The limitation "as such" is essential as Article 52(3) European Patent Convention (EPC) says that the exclusions to patentability that are listed in Article 52(2) EPC (including the exclusion for "programs for computers") only apply if the European patent (application) relates to any such excluded subject-matter *as such*. So, a valid European patent can be obtained as soon as a software-related invention relates to something that is not just a computer program *as such*. The Boards of Appeal of the European Patent Office have interpreted this by declaring an invention patentable as soon as there is a non-obvious technical contribution.

implementation that has similar functionality. According to patent law this independent implementation may also constitute an infringement. Depending on the scope of the patent, this might render it impossible to implement certain functionality without infringing a patent, which could mean that the software cannot be distributed and used according to open source principles[A15]. For these reasons, patents are not very popular in the open source community. The supporters of the Free Software Foundation, in particular, lobby very actively against software patents.

A3.3 Risks when using Open Source Software

Although the use of open source software may have clear advantages, there are also some risks. First of all, the software is made available without any kind of warranty or guarantees. Support is usually only available on a voluntary basis. Adhering to the license may be difficult for companies who are used to commercial licensing terms. The most important risks, however, are that under certain circumstances an application containing open source software has to be made available as open source software if it is put on the market, and that patents related to that application may become worthless. These risks mainly occur when the open source software in question is licensed under the GPL. Misusing an OSS license can lead to a lawsuit against your company and/or serious damage to brand equity.

Managing these risks can also have consequences at the architectural level and/or at the business level (patents). These two aspects are covered in the next sections.

A3.3.1 The architectural level

The GPL permits distribution of products based on software developed under it, but only if it is also distributed according to the terms of the GPL. There is unfortunately no precise definition of “based on”. The straightforward case is when source code from a work under the GPL is copied into another application. Statically linking several modules together to create an application would also seem to be covered. Dynamically linking two modules together is, however, quite controversial and, by using technologies such as CORBA, RPCs, sockets or plug-in modules, situations can rapidly become quite complex.

There is no clear consensus on the interpretation of the GPL. An often-heard compromise is to assume that any form of linking invokes the sublicensing clause, but using inter-process communication does not. It should however, be noted that using such technologies merely to avoid the consequences of the GPL would not be appreciated by the courts.

The Library GPL and the MPL are less problematic. It is permitted to create a “larger work” by combining the work under the MPL or LGPL with other software. This larger work may then be distributed under any licensing terms, as long as the conditions of the MPL or LGPL are adhered to for the portions of the larger work that were originally obtained under those licenses. This means that the source code of these parts must be made available – including any modifications – and the authors of these parts must be credited in the documentation of the larger work.

A3.3.2 The business level (patent risks)

- **Patent risks for users of open source software**
  Because of the large number of potentially relevant patents and their supposed triviality, many authors of open source software do not pay any attention to patents when writing their software. The chance that a particular piece of open source software infringes on some patent is therefore quite real.

  A patent holder can make life very difficult for the author of an open source software package. Of course this also holds for authors of commercial software, but open source authors usually do not have the means to take a license for all users of their software. Neither do they have a patent portfolio to improve their bargaining position. The free availability of the source code of course makes proving infringement very easy. However, it is very rare for a patent holder to approach the author directly.

  Distributors of CD-ROMs containing open source software regularly face claims from patent holders. Such distribution of infringing software usually counts as indirect infringement at the very least. A recent example is the Linux distributor Red Hat who has removed all MP3 software from its

[A15] In an economy where networking and interoperability are key at worldwide level, many companies are considering it is very important to develop shared standards. From a business standpoint, it may be of huge interest for a company, even if its policy is to develop a broad portfolio of patents for its core activity, to prefer the transformation of one of its inventions into a worldwide standard rapidly expanded through the open source movement, than to protect it by a patent, and to run the risk that the invention will be used in a limited way or even never.

distribution because of potential conflicts with the MP3 licensing program of the Fraunhofer Institute and Thomson multimedia.

The patent holder can of course, also approach users of infringing open source software. This is mainly a risk if the open source software has been incorporated in an expensive product or service, for example, as part of the operating system of a digital television or as part of the content management system for an online newspaper.

- **Risks for patent holders**
  The use of open source software can also introduce risks for patent holders themselves. Some open source software licenses contain provisions designed to reduce the risks of patents for authors and users of the software. If patent holders distribute open source software under such a license, they may be forced to grant a royalty-free license or a non-assert declaration to all users of this software. For example, anyone who contributes to a project licensed under the Mozilla Public License is required to give an unlimited, royalty-free patent license on such contributions worldwide. Users of software under this license can only assert patents they have on this software against the author(s) or other users if they also pay a reasonable royalty for their past use of the software. If no agreement can be reached on what constitutes “reasonable”, the license agreement terminates retroactively.

  The GNU General Public License (GPL) forbids imposing any restriction whatsoever on the rights granted by the license to the recipients of the software. If distributors do impose any further restrictions, their license under the GPL is cancelled. This means that patent holders who distribute a software package incorporating their patent can no longer assert that patent against those who distribute that package further or incorporate the package in their own product. Asserting a patent restricts the rights granted by the GPL and is therefore not permitted. This means that competitors are free to incorporate that package in their own product without having to pay any royalty to the patent holder. Of course that component of the product – and all other parts based on it – will have to be made available under the GPL.

  It does not matter whether the patent covers a contribution that the patent holder made to the software or the original version developed by someone else. The mere distribution in unmodified form of third party software under the GPL means that any patent covering that software can no longer be used against somebody using that software .

  The Library GPL contains a comparable provision, but restricted to patents covering the library itself. Patents that cover an application using the library do not have to be licensed freely.

  The latest potential weapons in the anti-patent arsenal are the Academic Free License and the Open Software License. These licenses contain a clause that automatically revokes the license if the licensee asserts any patents against any open source software package that is licensed under any license containing the same clause. If this clause becomes widely adopted, as its authors hope, it will become virtually impossible for patent holders to declare any patents against any open source software package.

A3.4 Conclusion

Using open source software offers various advantages, such as the ability to reduce costs and development time, and to avoid being dependent on a single vendor. It is therefore to be expected that more and more companies and institutions will start using open source software. There are, however, certain risks associated with doing so. Being forced to release some or all of the software of a commercial product as open source software may greatly reduce its value. And the risk to one’s patent portfolio should not be underestimated.

It is therefore recommended to *carefully study the license agreement and to make an assessment of the risks associated with these conditions*. One should always check whether one’s own application is clearly separated from software under the GPL. *Open source software should also be screened for*

patent risks before use. The chance that a particular open source software package infringes on a patent is substantial.

Furthermore it is recommended to compare the open source software with one's own patent portfolio. If there is a chance that one's own patent is incorporated in an open source software package, then the use of that software package should be avoided if the applicable open source software license is a patent-hostile license like the GPL, and if keeping the possibility to exploit the patent is deemed to be more valuable than being able to use the open source software package concerned. And, of course, compliance with the license conditions should always be checked.

With careful application of the license conditions, it is possible to gain maximum benefit from using open source software, while minimizing the risks.

A4 References

[Anderson 2003]
Ross Anderson - Trusted Computing' Frequently Asked Questions (TC / TCG / LaGrande /NGSCB / Longhorn / Palladium / TCPA), Version 1.1, August 2003; *http://www.cl.cam.ac.uk/~rja14*

[Audoin 2002]
Pierre Audoin Consulting – Application Software Products & Solutions Market in Western Europe December 2002

[Garvin 1984]
D.A. Garvin, Japanese Quality Management, Columbia Journal of World Business, 1984

[Feller Fitzgerald 2002]
Joseph Feller, Brian Fitzgerald, Understanding the Open Source Software Development, Pearson Education Limited, London 2002

[Kenwood 2001]
Carolyn A. Kenwood, A Business Case Study of Open Source Software, 2001
http://www.mitre.org/support/papers/tech_01/kenwood_software

[IHT 2003]
International Herald Tribune, "Microsoft sticks to tough tactics", Thursday 15 May, 2003

[Meta Group MySQL 2003]
Meta Group, MySQL Open Source Essentials, report, March 2003

[Meta Group EO 2003]
Meta Group, The Open Enterprise, report, 2003

[OSQ 2003]
Open Software Quality portal page: *http://osq.cs.berkeley.edu*

[Raymond 2001]
Eric S. Raymond, The Cathedral and the Bazaar, Musings on Linux and Open Source by an accidental Revolutionary, O'Reilly, January 2001.

[Reasoning 2003].
Reasoning Inc. How Open-Source and Commercial Software Compare: A Quantitative Analysis of TCP/IP Implementations in Commercial Software and in the Linux Kernel, 2003; *http://www.reasoning.com*

[Schmidt Porter 2001]
Douglas C. Schmidt, Adam Porter, Leveraging Open-Source Communities to improve the Quality & Performance of Open-Source Software; *http://www.cs.wustl.edu/~schmidt/PDF/skoll.pdf*

[StreamingMedia 2002]
New Eclipse Project Addresses Automated Software Quality-Open-Source;
http://industry.java.sun.com/javanews/stories/story2/0,1072,49456,00.html

A5 Glossary

ADAE	Agence pour le Développement de l'Administration Electronique
BSD	Berkeley Software Distribution
COTS	Commercial-Off-The-Shelf
CSS	Closed Source Software
CVS	Concurrent Versions System
DBMS	Data Base Management System
DoD	The US Department of Defense
DRM	Digital Rights Management
FLOS	Free/Libre and Open Source Software
FSF	Free Software Foundation
GPL	General Public License
HIS	Human System interaction
IPR	Intellectual Property Rights
ISV	Independent Software Vendor
KLOC	Thousands (Kilo) of Lines Of Code
KM	Knowledge Management
LGPL	Lesser/Library GPL (General Public License)
MZL	Mozilla Public License
OSS	Open Source Software
PCA	Project Cooperation Agreement
PGP	Pretty Good Privacy
ROI	Return On Investment
RPC	Remote Procedure Call
SI	System Integrator
SOAP	Simple Object Access Protocol
SSL	Secure Sockets Layer
TCP/IP	Transmission Control Protocol / Internet Protocol
UDDI	Universal Description Discovery & Integration
WSDL	Web Services Description Language
XML	eXtended Markup Language

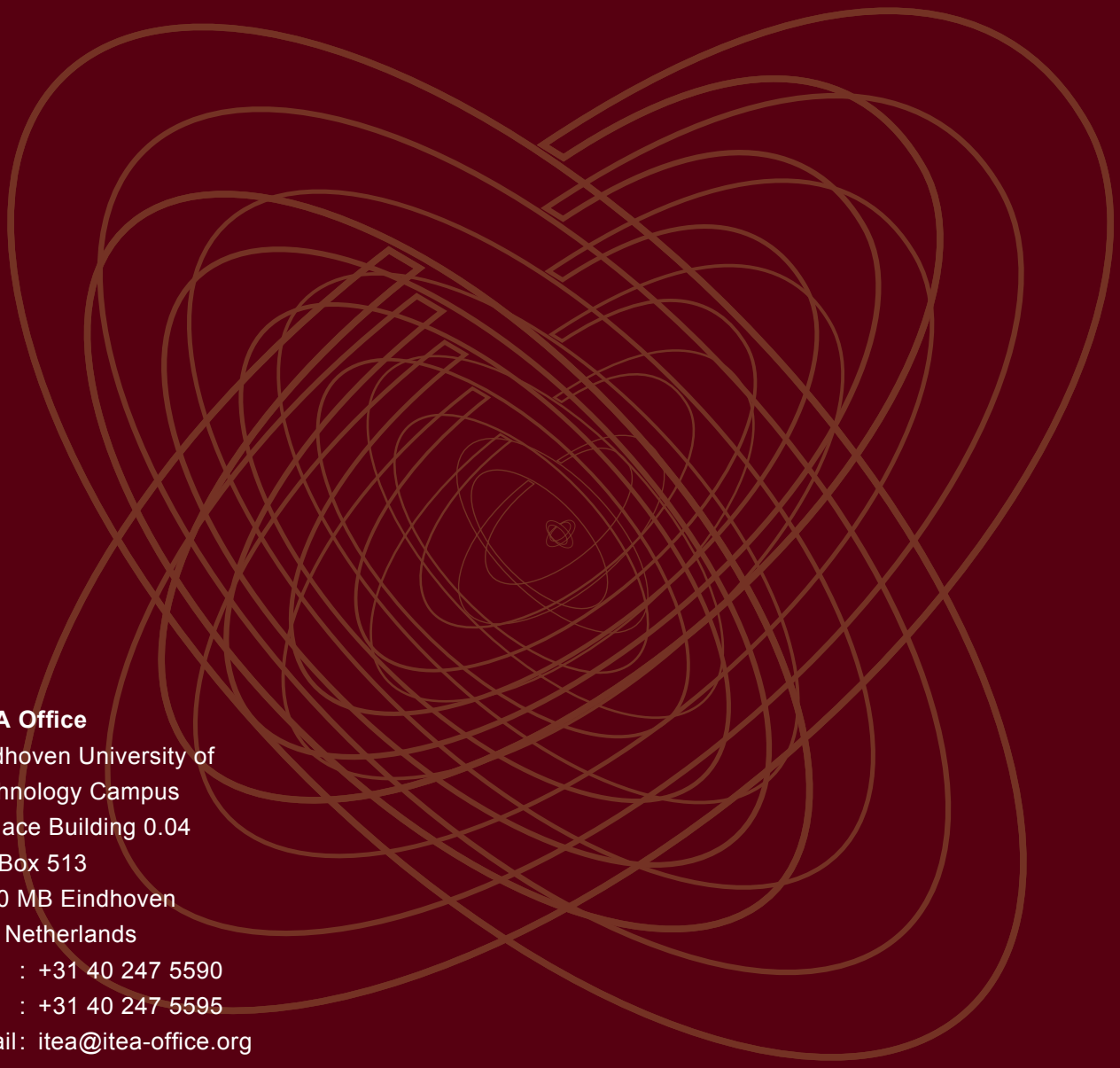
A6 Members of the ITEA Open Source Software Working Group

Coordinator:

Eric Daclin	*itea@itea-office.org*

Members:

Pierre Ficheux	*pierre.ficheux@openwide.fr*
Jean-Pierre Laisne	*jean-pierre.laisne@bull.net*
Jacques Leroudier	*jacques.leroudier@thomson.net*
Winfried Seidel	*winfried.seidel@siemens.com*
Tapio Tallgren	*tapio.tallgren@nokia.com*
Jean-Michel Tanneau	*jean-michel.tanneau@thalesgroup.com*
Ger Verkoeijen	*ger.verkoeijen@philips.com*
Matthias Weber	*matthias.n.weber@daimlerchrysler.com*



ITEA Office
Eindhoven University of
Technology Campus
Laplace Building 0.04
PO Box 513
5600 MB Eindhoven
The Netherlands
Tel : +31 40 247 5590
Fax : +31 40 247 5595
Email: itea@itea-office.org
Web : www.itea-office.org

ITEA

INFORMATION TECHNOLOGY

FOR EUROPEAN ADVANCEMENT

ITEA is a EUREKA strategic
cluster programme

Σ! 2023